HOCHSCHULE
MITTWEIDA
UNIVERSITY OF
APPLIED SCIENCES

BACHELORARBEIT

Herr
Hannes Hartung

Untersuchung von Smartcards mit NFC-Chips

2017

Fakultät: Angewandte Computer- und Biowissenschaften

BACHELORARBEIT

Untersuchung von Smartcards mit NFC-Chips

Autor:
Herr Hannes Hartung

Studiengang:
Allgemeine und Digitale Forensik

Seminargruppe:
FO14w2-B

Erstprüfer:
Prof. Dr. rer. pol. Dirk Pawlaszczyk

Zweitprüfer:
Prof. Dr. rer. nat. Christian Hummert

Faculty of Applied Computer Sciences and Biosciences

BACHELOR THESIS

Scientific review of smartcards with nfc technology

author:
Mr. Hannes Hartung

course of studies:
General and Digital Forensics

seminar group:
FO14w2-B

first examiner:
Prof. Dr. rer. pol. Dirk Pawlaszczyk

second examiner:
Prof. Dr. rer. nat. Christian Hummert

Bibliografische Angaben

Hartung, Hannes:

Untersuchung von Smartcards mit NFC – Chips

Scientific review of smartcards with nfc technologoy

90 Seiten, Hochschule Mittweida, University of Applied Sciences,
Fakultät Angewandte Computer- und Biowissenschaften, Bachelorarbeit, 2017

Abstract

Diese Arbeit beschäftigt sich mit Smartcards, welche mit Hilfe der NFC-Technologie kontaktlos Daten übertragen können. Zunächst wurde der theoretische Rahmen gespannt, um die Grundsteine dieser Forschung zu erläutern: RFID, Smartcards und NFC. In einer explorativen Untersuchung fand die Analyse und der Vergleich von Smartcards verschiedener Hersteller statt. Dazu wurden diese in einer virtuellen Testumgebung mit unterschiedlichen Android-Applikationen beschrieben und ausgelesen. Als Referenzanwendung diente eine selbstprogrammierte App. Die Ergebnisse zeigen, dass es ohne große Fachkenntnisse und mit geringem zeitlichen und finanziellen Aufwand möglich ist, Smartcards mit NFC-Chips auszulesen, welche für sensible Anwendungen, wie beispielsweise kontaktlose Bezahlvorgänge oder Patientendaten, verwendet werden.

Inhaltsverzeichnis

Abkürzungsverzeichnis

PCD – Proximity Coupling Device

PICC – Proximity Integrated Circuit Card

NFC – Near Field Communication

ISO – International Organization for Standardization

UID - Unique Identifier

DES – Data Encryption Standard

PKE – Public Key Encryption

RFID – Radio-Frequency Identification

LLCP – Logical Link Control Protocol

MAC – Media Accesss Control Layer

LLC – Logical Link Control Layer

EEPROM - Electrically Erasable Programmable Read-only Memory

NDEF – NFC Data Exchange Format

TNF – Type Name Format

MIME – Multipurpose Internet Mail Extensions

JPEG - Joint Photographic Experts Group

AEE – Application Execution Environment

TEE – Trusted Execution Environment

ICC – Integrated Circuit Card

MITM – Man In The Middle

SSE – Shared Secret Service

SCH - Secure Channel Service

EDCH - Elliptic Curves Diffie-Hellman

IDE – Integrated Development Environment

SDK – Software Development Kit

ADT – Android Development Tools

API – Application Programming Interface

URI – Uniform Ressource Identifier

IEEE- Institute of Electrical and Electronics Engineers

URN – Uniform Ressource Name

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

1.1 Motivation

In der heutigen Zeit geht es in allen Bereichen der Technik um Prozessoptimierung und Effizienzsteigerung. Eine Technik, die durch ihre einfache Bedienung und die unzähligen Anwendungsbereiche besticht, ist die „Near Field Communication“ (NFC). Weltweit nutzen Millionen von Menschen täglich NFC-fähige Smartcards und Smartphones, um im Supermarkt zu bezahlen, Kino- und Konzertkarten zu erwerben, Fotos oder andere Medien zu übertragen sowie als Fahrkarte im öffentlichen Personennahverkehr. Nicht nur der Endnutzer profitiert davon. In der Industrie übernimmt NFC mittlerweile flächendeckend Aufgaben wie: die Zutrittskontrolle, die Zeiterfassung und die Datenübertragung für z.B. Wartungssysteme. Ein weiterer interessanter Einsatzbereich für NFC ist die Medizin. Anwendungen, um Patientendaten wie Blutdruck- oder Blutzuckermessungen auszulesen, die Authentifizierung mithilfe der Krankenkassenkarte oder die Speicherung und Übertragung relevanter Daten der Krankenakte. Täglich kommen neue Anwendungen dazu, um Prozesse zu optimieren, Dienstleistungen schneller abzuwickeln oder die Bedienung für den Endnutzer zu erleichtern. Aber wo die Benutzerfreundlichkeit und der Komfort steigt, bedeutet das meist einen Verlust an Sicherheit. Bietet NFC dem Nutzer zusätzlich zu der einfachen Bedienung auch eine sichere Kommunikationsebene? Durch die Untersuchung von NFC-fähigen Smartcards möchte der Autor die beschriebene Technologie untersuchen und Klarheit schaffen.

1.2 Aufbau der Arbeit

Die vorliegende Arbeit ist folgendermaßen aufgebaut. In Kapitel 2, „Theoretische Grundlagen der Arbeit“, widmet sich der Autor den zentralen Bausteinen dieses Forschungsprojekts. Der theoretische Teil folgt einem strukturierten Aufbau. Den Aufbau der Grundlagenerklärung von Begrifflichkeiten nimmt der Autor folgendermaßen vor. Zuerst wird die Begriffsherkunft und die Übersetzung in die deutsche Sprache vollzogen. Im Anschluss widmet sich der Autor einer kurzen Definition. Danach stellt er die historische Entwicklung der Thematik vor und zuletzt wird der Bezug zur vorliegenden Arbeit hergestellt. Kapitel 2.1 beschäftigt sich mit Smartcards und erklärt unter anderem den Unterschied zwischen Speicher- und Prozessorkarten. Weiterhin werden die wichtigsten Kommunikationsschnittstellen und

Übertragungsprotokolle vorgestellt. Im Anschluss nennt der Autor die wichtigsten Smartcardtypen und deren Hersteller und außerdem, welche Normen für die Herstellung berücksichtigt werden müssen. Der letzte Unterpunkt beschäftigt sich mit der derzeitigen Sicherheitssituation von Smartcards und beleuchtet Risiken, die in der IT-Sicherheit an der Tagesordnung sind. Kapitel 2.2 beschäftigt sich mit RFID. Diese Technologie ist die Basis, auf der die NFC–Technologie aufbaut. Um ein grobes Verständnis für die Entwicklung zu erlangen, wird die RFID-Technologie in ihren Grundzügen erläutert. Kapitel 2.3 beschäftigt sich detailliert mit der NFC-Technologie. Da diese ein zentraler Bestandteil dieser Arbeit ist, werden Anwendungsgebiete, Kommunikationsmodi und Datenformate erläutert. Auch hier bildet den Abschluss eine Sicherheitsbetrachtung, um NFC in den Kontext der IT-Sicherheit einzuordnen. Kapitel 3 zeigt exemplarisch die Herangehensweise zur Programmierung einer NFC–Applikation für das Betriebssystem Android, um Smartcards mit NFC-Chips zu untersuchen. Angefangen vom Aufsetzen der Entwicklungsumgebung über Basiskonfigurationen und Operationsmodi wird alles erläutert. In Kapitel 4 widmet sich der Autor der Analyse und dem Forschungskonzept des vorliegenden Forschungsvorhabens. Dazu wird zunächst der Bedarf ergründet. Im Anschluss wird gezeigt, welchen Ansatz diese Arbeit verfolgt und wie das methodische Vorgehen gestaltet ist. Das letzte Kapitel dient der Auswertung und Evaluation der Ergebnisse. Der Autor beantwortet die zuvor gestellten Forschungsfragen, interpretiert die Ergebnisse und gibt einen Ausblick. Zum Schluss werden die wichtigen Aspekte dieser Arbeit zusammengefasst und ein Fazit gezogen. Am Ende findet sich das Literaturverzeichnis.

1.3 Ziel der Arbeit

Das Ziel der Forschung ist es, ein in der Praxis beobachtetes Phänomen wissenschaftlich greifbar zu machen. Bei dem Phänomen handelt es sich um die Verwendung von Smartcards mit NFC-Chips. Diese Karten finden mittlerweile in unzähligen alltäglichen Situationen Anwendung. Der Autor möchte das Thema von der Basis an ergründen und jedem mit Grundkenntnissen der Informatik ermöglichen die Funktionen, Chancen, und auch Risiken zu verstehen. Zum einen soll dafür der theoretische Rahmen gespannt werden, um nur mithilfe dieser Arbeit Smartcards mit NFC-Chips besser verstehen zu können und weiterhin selber Untersuchungen anstellen zu können. Ziel ist es weiterhin eine Basis für zukünftige Forschungsprojekte zu schaffen, die das Phänomen weiter ergründen und weitere Erkenntnisse sammeln

2 Theoretische Grundlagen der Arbeit

2.1 Smartcards

Der Begriff Smartcard stammt aus dem Englischen und setzt sich aus den beiden Begriffen „smart“ und „card“ zusammen. Übersetzt bedeutet das „schlaue Karte“. Weitere Begriffe für die Smartcard sind „Chipkarte“ oder „Integrated Circuit Card“ (ICC). Übersetzt bedeutet ICC „Karte mit integriertem Schaltkreis“.

Im Allgemeinen versteht man unter Smartcards Kunststoffkarten, die mit einem Mikrochip ausgestattet sind. Die Maße einer Chipkarte sind international genormt. [1]

Unter dem Werbeslogan „Bezahlen Sie einfach mit Ihrem guten Namen!“ führte Anfang der 1950er Jahre die Firma Diners Club die erste Plastikkarte als Zahlungsmittel ein. Zusätzlich zu dem normalen Aufdruck wurde die Karte mit einer Hochprägung und einem Unterschriftsbereich ausgestattet. Die Hochprägung enthielt einige persönliche Informationen. Damals galten diese Mechanismen als ausreichende Sicherheitsmerkmale. [2] [3]

Im Jahre 1968 erfolgte die Patentanmeldung für Identifikationskarten mit integriertem Schaltkreis.[4] Erst „in den 1970er Jahren gelang es, Rechnerlogik und Datenspeicher auf einem einzigen Siliziumplättchen mit wenigen Quadratmillimetern Fläche zu inte-grieren.“ (Langer und Roland 2010, S. 4) Die Entwicklung und Ausbreitung des Einsatzgebietes von Chipkarten erfolgte parallel zur immer mehr eingesetzten elektronischen Datenverarbeitung. Der erste große kommerzielle Einsatz erfolgte 1984 in Frankreich. Die französische Firma „Poste, Téléphone et Télécommunications“(PTT) führte Chipkarten für Telefone ein. Im Jahr 1990 waren bereits 60 Millionen und im Jahr 1997 mehrere hundert Millionen weltweit im Umlauf. [5] Der nächste große Schritt in der Geschichte der Smartcard war die Implementierung von

[1] Vgl. Langer und Roland 2010, S. 33

[2] Vgl. Holger Schinzel 1999, S.44

[3] Vgl. Langer und Roland 2010, S. 1–2

[4] Vgl. Brenner und Kolbe 2012, S. 64

[5] Vgl. Rankl und Effing 2008, S. 2–8

Mikroprozessoren im Zusammenspiel mit Kryptocoprozessoren auf einer Karte. Als sogenannte „Geldkarte“ kam dieses System erstmals 1996 in Deutschland zum Einsatz. [6]

Die Smartcard spielt für diese Arbeit eine entscheidende Rolle. Heutzutage existiert eine Vielzahl von verschiedenen eingesetzten „Chips“. Die Auswertung und der Vergleich verschiedener Chipkarten ist somit Bestandteil dieser Arbeit.

Smartcards lassen sich anhand verschiedener Ansätze klassifizieren. In den nächsten beiden Kapiteln wird eine Unterteilung in Speicher- und Prozessorkarten vorgenommen. Während Speicherkarten nur aus einem Speicher und einfachen Zugriffslogiken bestehen, haben Prozessorkarten einen Mikrocontroller. Dieser führt die kryptografischen Operationen aus, verwaltet das Dateisystem und arbeitet die gewünschten Anwendungen ab. [7]

2.1.1 Speicherkarten

Die Gruppe der Speicherkarten kann wiederum in Speicherkarten mit und Speicherkarten ohne Sicherheitslogik unterteilt werden. Bevor näher auf diese Unterscheidung eingegangen wird, zeigt Abbildung 1 den Basisaufbau einer Speicherkarte.

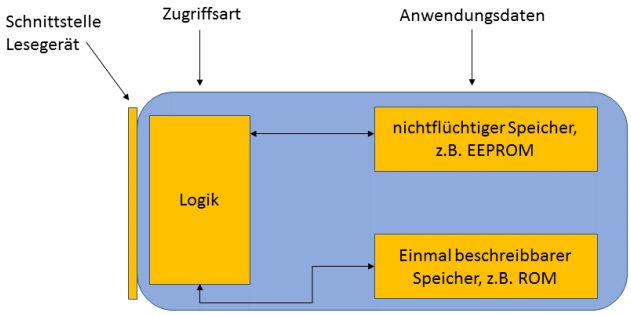

Abbildung 1: Basisaufbau Speicherkarte

[6] Vgl. Langer und Roland 2010, S. 4

[7] Vgl. Langer und Roland 2010, S. 34

Das Hauptelement der Speicherkarte ist ein nichtflüchtiger Speicher. Bei Karten ohne Sicherheitslogik wird zumeist ein EEPROM-Speicher verwendet. In späteren Kapiteln wird diese Speicherform genauer erläutert. Ein weiterer Bestandteil ist ein einmal beschreibbarer Speicher. Der Zugriff auf diesen wird von der Zugriffslogik oder der Sicherheitslogik verwaltet.

Speicherkarten ohne Sicherheitslogik ermöglichen einen freien Zugriff auf den Datenspeicher. Das bedeutet, dass dieser frei lesbar, beschreibbar und löschbar ist. Speicherkarten mit Sicherheitslogik schützen gewisse Bereiche und Funktionen. Dabei kann der Lese- oder Schreibzugriff oder ein Teil des Speichers geschützt bzw. erst nach Authentifizierung freigegeben werden. Der Stand der Technik heutzutage beinhaltet aufwendigere kryptografische Verfahren, um die gewünschten Schutzbereiche abzusichern. Durch das Zusammenspiel der Logik und des Speichers wird die Funktionalität der Karte gewährleistet. Dabei werden einfache Adressierungen, aber auch aufwendige Zustandsautomatismen eingesetzt. [8]

2.1.2 Prozessorkarten

Abbildung 2 veranschaulicht den Aufbau einer Prozessorkarte. Das Hauptelement dieser Karte ist der Prozessor. Um die Daten und Anwendungssoftware abzulegen, sind mehrere Speichertypen implementiert. Das Betriebssystem und die Anwendungen werden im EEPROM-Speicher und auf dem ROM-Speicher abgelegt. Die Daten und Informationen werden auf dem RAM gespeichert. Zusätzlich ist eine Memory-Management-Unit (MMU) implementiert, die alle Speicherzugriffe überblickt. Ein weiterer entscheidender Vorteil gegenüber den Speicherkarten ist die Sicherheit. Bei Prozessorkarten sind zusätzliche Co-Prozessoren, wie z.B. ein Kryptokoprozessor implementiert, um auch sicherheitsrelevante Anwendungen geschützt ausführen zu können. Weiterhin existiert ein „Cyclic Redundancy Check Generator“ (CRC). Dieser ist für die Verarbeitung der Daten zuständig und berechnet Prüfsummen. Um mit Lesegeräten oder anderen Geräten zu kommunizieren, enthält die Prozessorkarte verschiedene Schnittstellen. Alle Schnittstellen bestehen aus zwei Teilen, einem „Universal Asynchronous Receiver and Transmitter“ (UART) und einem analogen Teil. Da die Angriffsmöglichkeiten bei Prozessorkarten vielseitig sind und es mehrere Bereiche gibt, die als sicherheitskritisch anzusehen

[8] Vgl. Langer und Roland 2010, S. 35

sind, werden meist Sicherheitssensoren implementiert, um die verschiedenen Abschnitte bewachen zu können. [9]

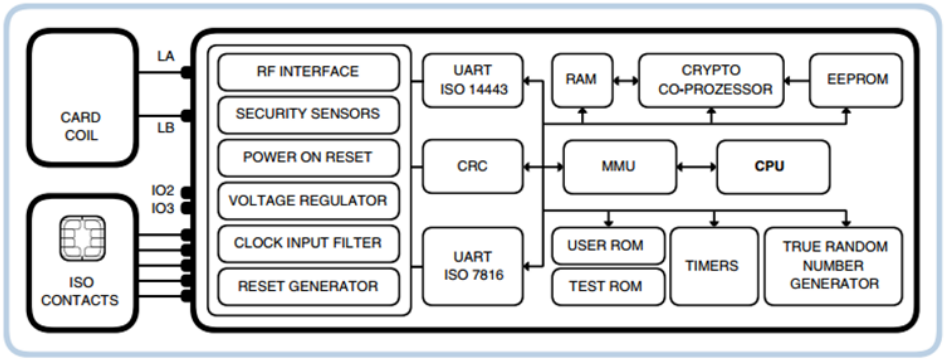

Abbildung 2: Basisaufbau Prozessorkarte Quelle: (Langer und Roland 2010, S. 61)

2.1.3 Kommunikationsschnittstellen

Wie in den letzten Abschnitten beschrieben wurde, existieren zur Kommunikation mit Lese- und anderen Geräten sowohl bei Speicherkarten als auch bei Prozessorkarten verschiedene Kommunikationsschnittstellen. Man unterscheidet dabei in kontaktbehaftete und kontaktlose Schnittstellen. Da diese Arbeit sich vorwiegend mit den kontaktlosen Verfahren beschäftigt, liegt das Augenmerk der Betrachtung auch auf diesen Schnittstellen. Kontaktbehaftete Schnittstellen nehmen die Einteilung in Prozessorkarten und Speicherkarten vor. Für die Speicherkarten existieren synchrone Übertragungsprotokolle und für die Prozessorkarten gibt es eigene Protokolle für die Übertragung von Daten. Die Protokolle sind genormt. Der Standard ISO/IEC 7816 beschäftigt sich damit. Im folgenden Abschnitt wird darauf genauer eingegangen.

Im Rahmen der Recherche wurden viele kontaktlose Schnittstellen ausfindig gemacht. Eine sehr gute Übersicht der wichtigsten enthält das Buch „Anwendungen und Technik von NFC“, welches von Michael Roland und Joseph Langer verfasst wurde. (siehe Abb. 3)

[9] Vgl. Langer und Roland 2010, S. 36

Typ	Reichweite	Norm
Close Coupling	ca. 1 cm	ISO/IEC 10536 [11–13]
Proximity Coupling	ca. 10 cm	ISO/IEC 14443 [14–17]
FeliCa	ca. 10 cm	JIS X 6319-4 [24]
Vicinity Coupling	ca. 1 m	ISO/IEC 15693 [18–20]
EPCglobal UHF Class 1 Generation 2	ca. 10 m	ISO/IEC 18000-6C [21, 22]

Abbildung 3: Übersicht Schnittstellen Quelle: (Langer und Roland 2010, S. 38)

Die verschiedenen Schnittstellen unterscheiden sich in ihrem Funktionsumfang, der Betriebsfrequenz, der zugrundeliegenden Norm und der Reichweite. Je nach gewünschter Anwendung muss die Schnittstelle passend ausgewählt werden. Da die meisten Karten passiver Natur sind, muss es über die Schnittstelle möglich sein, Versorgungsenergie zu transferieren. Systeme, die auf induktiver Kopplung basieren und auf einer Betriebsfrequenz von 13,56 MHz agieren, sind die Basis für die sogenannte Nahfeldkommunikation (NFC). Genauer wird darauf im Kapitel NFC eingegangen. Beispiele für diese Schnittstellen sind „Proximity Coupling“, „FeliCa“ und „Vicinity Coupling“. (siehe Abb. 3) [10]

2.1.4 Übertragungsprotokolle

Kommunikationspartner für die Übertragung sind zumeist Lesegeräte und eine oder mehrere Smartcards. Übertragungsprotokolle regeln die genaue Struktur und Vorgehensweise dieser Kommunikation. Ein Modell, was diese Art der Struktur besonders gut beschreibt, ist das OSI-Schichtenmodell. [11]

[10] Vgl. Langer und Roland 2010, S. 36–38

[11] Vgl. Prosser 1993, S. 3

Dieses besteht aus sieben Schichten, in die Kommunikationsprotokolle eingeordnet werden können:

1. Bitübertragungsschicht
2. Sicherungsschicht
3. Vermittlungsschicht
4. Transportschicht
5. Steuerungsschicht
6. Darstellungsschicht
7. Anwendungsschicht

Für die Kommunikation von Smartcards sind vor allem vier Ebenen wichtig. (siehe Abbildung 4)

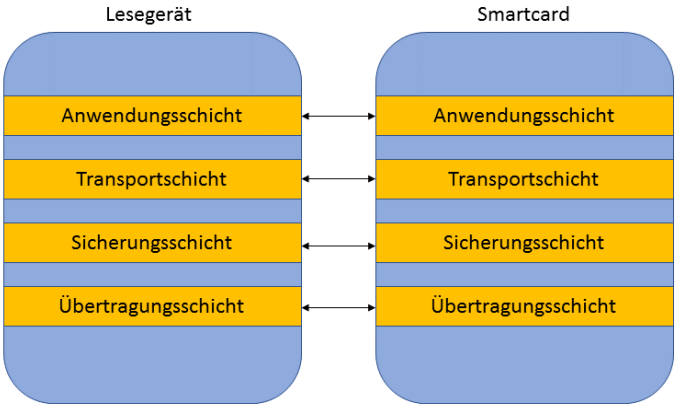

Abbildung 4: Protokollstapel Smartcards

In der nachfolgenden Tabelle werden die einzelnen Funktionen der Ebenen genauer beschrieben.

Schicht	Funktion
Übertragungsschicht	Signalpegel- und Form für Datenaustausch
Sicherungsschicht	Fehlerkorrektur, Erzeugen der Chipkarte, Antikollision
Transportschicht	Transport der Pakete
Anwendungsschicht	Übertragung von Befehlen der Anwendung

Tabelle 1: Funktionen der Schichten

Für Smartcards im Allgemeinen existieren eine große Anzahl an Übertragungsprotokollen. Das liegt unter anderem daran, dass für die verschiedenen Typen auch unterschiedliche Anforderungen gegeben sind. [12]

2.1.5 ISO/IEC 14443

Für kontaktlose Chipkarten ist die Arbeitsgruppe WG8 der internationalen Organisation für Normung (ISO) zuständig. Die Norm ISO/IEC 14443 ist der Standard für kontaktlose Chipkarten und beschreibt die Kommunikation und Abläufe, die zwischen Chipkarten und Lesegeräten stattfinden. Die Norm beschreibt die Leseeinheit als „Proximity Coupling Device“ (PCD) und die Chipkarte als „Proximity Integrated Circuit Card“ (PICC). Die Kommunikationsreichweite wird mit sieben bis fünfzehn Zentimetern beschrieben. Die Norm ist in mehrere Teile gegliedert. Der erste befasst sich mit den physischen Eigenschaften der PICCs.

[12] Vgl. Langer und Roland 2010, S. 43–46

Die maximale Größe der Antenne ist mit 86 mm * 54 mm * 3 mm festgelegt. (siehe Abb.5)

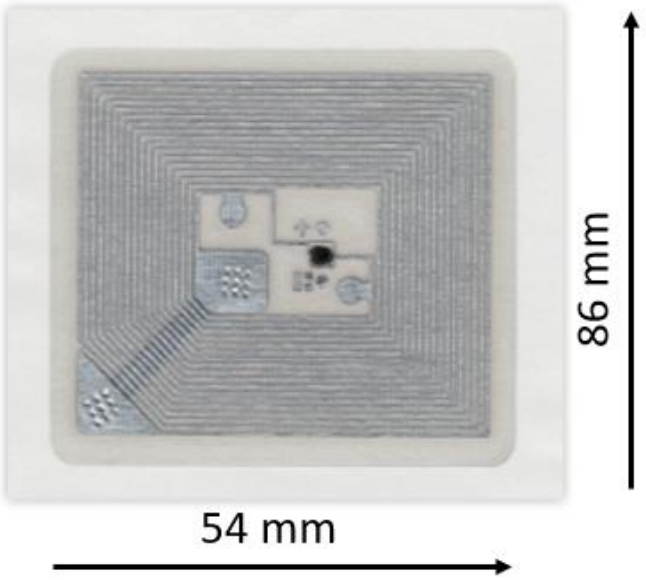

Abbildung 5: NFC-Antennenmaße (Quelle: https://www.generationrobots.com/de/401709-nfc-shield-v20.html)

Für weitere Eigenschaften wird auf die Normen ISO/IEC 7810 und ISO/IEC 15457-1 verwiesen. Alle Maße und Materialien sind so zu wählen, dass der Frequenzbereich bei 13,56 MHz, bei einer Maximalbelastung von 12 A/m für ca. 30 Sekunden gewährleistet wird.

Der zweite Teil der Norm beschäftigt sich mit der Signalerkennung und Energieversorgung und legt diese fest. Der PCD erzeugt ein Magnetfeld mit einer Sendefrequenz von 13,56 MHz und versorgt so die PICC mit Energie. Die magnetische Feldstärke muss im Bereich von 1,5 A/m und 7,5 A/m liegen. Die minimale Feldstärke von 1,5 A/m ist sehr wichtig, da für die PICC keine aktive Stromquelle existiert. Somit sollte die minimale Feldstärke bei einem Abstand von 0-10cm mindestens zur Verfügung stehen, um die Kommunikation zu gewährleisten. Als Nächstes werden die Modulationsverfahren für die Typen A und B festgelegt.

Der dritte Teil der Norm befasst sich mit der Kommunikation und dem Datenaustausch zwischen den beteiligten Geräten. Der letzte Teil beschreibt das Übertragungsprotokoll für den kontaktlosen Datenaustausch. [13]

2.1.6 Smartcardtypen- und hersteller

Einer der größten Hersteller von Smartcards ist die Firma NXP Semiconductors mit Hauptsitz in den Niederlanden. Mit einer großen Produktvielfalt schafft diese Lösungen für kontaktlose Zugangs- und Zeitkontrollen und sichere kontaktlose Automobilzugangskontrollen. Die weltweit meistgenutzte kontaktlose Chipkartentechnik „MIFARE“ wurde von NXP Semiconductors entwickelt und die Produktfamilie umfasst mittlerweile vier Produktreihen. (siehe Abb. 6)

Product features	MIFARE Ultralight®			MIFARE Classic®		MIFARE Plus®							MIFARE®DESFire®						
	Nano	EV1	C	EV1		S		SE	X		EV1		EV1				EV2		
RF Interface	ISO/IEC 14443-2, Type A 13.56 MHz																		
Protocol	ISO/IEC 14443-3					ISO/IEC 14443-3&4							ISO/IEC 14443-4						
UID – unique identifier	7-byte UID			7-byte UID, 4-byte NUID, Random ID									7-byte UID, Random ID						
Communication speed	106 Kbps					106-848 Kbps													
Memory size [Bytes]	40	48	128	144	1K	4K	2K	4K	1K	2K	4K	2K	4K	2S6	2K	4K	8K	2K	4K	8K
Memory model	Compact, 4-byte page				Compact, Sectors & 16-byte block									Flexible file system						
Crypto				TDES	Crypto-1		Crypto-1, AES							DES / 2K3DES / 3K3DES / AES						
Key length				112-bit	48-bit Crypto-1		48-bit Crypto-1, 128-bit AES							128-bit AES, up to 168-bit DES						
Authentication		Password			3-pass mutual														
Communication security	-				Encrypted		Plain, CMACed, Encrypted w. CMAC							Plain, CMACed, Encrypted w. CMAC						
MIsmartApp	-													-				✓		
Transaction MAC	-						-					✓		-				✓		
Multi key sets	-													-				✓		
Proximity check	-									✓				-				✓		
Virtual card select	-						✓							-				✓		
Originality check features	ECC signature re-programmable	ECC signature		ECC signature		AES originality keys						AES originality keys, ECC signature						AES originality keys, ECC signature		
CC Certification	-					EAL4+		-	EAL4+			EAL5+		EAL4+				EAL5+		
ISO 7816-4 APDU	-											✓		✓						
NFC compliance	NFC Forum tag type 2 compliant			Not supported by majority of NFC devices		NFC capable in SL3						NFC capabilities in SL1 and SL3		NFC Forum Tag Type 4 V2.0 compliant						
Target applications	Public transport & event ticketing Loyalty programs, limited use tickets			Various applications – recommended to move to higher security ICs		Public transport / campus cards / access management								Smart city platform / advanced mobility multi-applications / micropayment / loyalty programs / access management						
Input capacitance [pF]	17 / 5			17 / 50		17						17 / 70		17 / 70						
Multi applications	-			supported via MAD		supported via MAD								dynamic						
Delivery types – 7 Byte UID																				
Wafer 120µm / 17 pF	MF0UN0001 DUD	MF0UL1101 DUD	MF0UL2101 DUD	MF0MCU2001 DUD	MF1S501X DUD[1]	MF1S7001 XDUD[1]	MF1SPLUS6001 DUD[1]	MF1SPLUS8001 DUD[1]	MF1SEP1001 DUD[1]	MF1PLUS6001 DUD[1]	MF1PLUS8001 DUD[1]	MF1P2101 DUD[1]	MF1P4101 DUD[1]	MF3ICDQ101 DUD	MF3ICD2101 DUD	MF3ICD4101 DUD	MF3ICD8101 DUD	MF3D2201 DUD	MF3D4201 DUD	MF3D8201 DUD
Wafer 120 µm / high cap	MF0UN0001 DUF	MF0UL1101 DUD	MF0UL2101 DUD	MF0MCU2101 DUD	-	-	-	-	-	-	-	MF1PH2101 DUD[1]	MF1PH4101 DUD[1]	MF3ICDHQ101 DUD	MF3ICDH2101 DUD	MF3ICDH4101 DUD	MF3ICDH8101 DUD	MF3DH2201 DUD	MF3DH4201 DUD	MF3DH8201 DUD
Wafer 75 µm / 17pF	MF0UNH0001 DUD	MF0UL1101 DUF	MF0UL2101 DUF	-	MF1S5001X DUF[1]	MF1S7001 XDUF[1]	-	-	-	-	-	MF1P2101 DUF[1]	MF1P4101 DUF[1]	MF3ICDQ101 DUF	MF3ICD2101 DUF	MF3ICD4101 DUF	MF3ICD8101 DUF	MF3D2201 DUF	MF3D4201 DUF	MF3D8201 DUF
Wafer 75 µm / high cap	MF0UNH0001 DUF	MF0UL1101 DUF	MF0UL2101 DUF	-	-	-	-	-	-	-	-	MF1PH2101 DUF[1]	MF1PH4101 DUF[1]	MF3ICDHQ101 DUF	MF3ICDH2101 DUF	MF3ICDH4101 DUF	MF3ICDH8101 DUF	MF3DH2201 DUF	MF3DH4201 DUF	MF3DH8201 DUF
MOA4 / 17pF	-	-	-	MF0MOU2001 DA4	-	-	MF1SPLUS6001 DA4[1]	MF1SPLUS8001 DA4[1]	MF1SEP1001 DA4[1]	MF1PLUS6001 DA4[1]	MF1PLUS8001 DA4[1]	MF1P2100 DA4[1]	MF1P4100 DA4[1]	-	MF3MOD2101 DA4	MF3MOD4101 DA4	MF3MOD8101 DA4	MF3D2200 DA4	MF3D4200 DA4	MF3D8200 DA4
MOA4 / high cap	-	-		MF0MOU2101 DA4	MF1S5000 XDA4[1]	MF1S7000 XDA4[1]	-	-	-	-	-	MF1PH2100 DA4[1]	MF1PH4100 DA4[1]	-	MF3MODH2101 DA4	MF3MODH4101 DA4	MF3MODH8101 DA4	MF3DH2200 DA4	MF3DH4200 DA4	MF3DH8200 DA4
MOA8 / 17 pF	-		MF0UL2101 DA8	MF0MOU2001 DA8	-	-	MF1SPLUS6001 DA8[1]	MF1SPLUS8001 DA8[1]	MF1SEP1001 DA8[1]	MF1PLUS6001 DA8[1]	MF1PLUS8001 DA8[1]	-	-	MF3MODQ101 DA8	MF3MOD2101 DA8	MF3MOD4101 DA8	MF3MOD8101 DA8	-	-	-
MOA8 / high cap	-	-	-	-	MF1S5000 XDA8[1]	MF1S7000 XDA8[1]	-	-	-	-	-	-	-	MF3MODHQ101 DA8	MF3MODH2101 DA8	MF3MODH4101 DA8	MF3MODH8101 DA8	-	-	-
MOB6 / 17pF	-	-	-	-	-	-	-	-	-	-	-	MF1P2100 DA6[1]	MF1P4100 DA6[1]	-	-	-	-	MF3D2200 DA6	MF3D4200 DA6	MF3D8200 DA6
MOB6 / high cap	-	-	-	-	-	-	-	-	-	-	-	MF1PH2100 DA6[1]	MF1PH4100 DA6[1]	-	-	-	-	MF3DH2200 DA6	MF3DH4200 DA6	MF3DH8200 DA6

[1] available also in legacy 4 Byte NUID

MIFARE, MIFARE Ultralight, MIFARE Classic, MIFARE Plus and MIFARE DESFire are registered trademarks of NXP B.V.

*Abbildung 6: Mifareproduktfamilie (*Quelle: https://www.nxp.com/docs/en/product-selector-guide/MIFARE_ICs_939775017001_v9_HR.pdf)

[13] Vgl. Jonas Groß 2012, S. 6-7

Alle MIFARE-ICs sind konform zur Norm ISO/IEC 14443 und erfüllen somit die Standards für die kontaktlose Kommunikation zwischen Chipkarten. Die MIFARE-Classic- Reihe ist die ursprüngliche Produktreihe. Diese Chipkarten sind in unzähligen Infrastrukturen implementiert. In London werden die MIFARE-Classic-Karten seit 2003 als Bezahlsystem für öffentliche Verkehrsmittel genutzt. [14] Mit dem Pay-As-You-Go Verfahren können Beträge auf die Karte aufgeladen werden. Die MIFARE-Classic-Technologie nutzt zur Kommunikation ein proprietäres, normkonformes High-Level-Protokoll. Die in diesen Karten verbauten Tags sind mit einem EEPROM-Speicher ausgestattet. Die Speichergrößen sind 320 Byte, 1 Kilobyte und 4 Kilobyte. Seit 2007 gilt die Karte als kritisch im Bereich der IT-Sicherheit.

Der verwendete Crypto1-Algorithmus wurde dort mehrmals erfolgreich angegriffen.[15] Die Karte ist außerdem standardmäßig mit einer 4 Byte langen Identifikationsnummer (UID) ausgestattet. Da sehr viele Karten in Umlauf kamen, ist seit 2010 der Punkt überschritten, wo UIDs mehrmals vergeben wurden. Deshalb wird bei späteren Standards auf eine 7 Byte lange UID gesetzt.

Die zweite Kategorie der Produktplatte sind die MIFARE- Ultralight- Karten. Diese Karten sind in zwei verschiedenen Versionen mit unterschiedlicher Speichergröße und Verschlüsselung erhältlich. Die Standard Ultralight-Version ist sehr günstig, da unverschlüsselt und mit einer Speichergröße von 512 Bit. Zur Kommunikation wird dasselbe Protokoll wie bei den Classic-Karten genutzt. Dieser Chip ist schon mit einer 7 Byte langen UID ausgestattet. Aufgrund der geringen Kosten wird dieser Chip oft bei Einmaltickets im Bereich des öffentlichen Verkehrs, im Event-Bereich oder bei Bonusprogrammen benutzt. Dort ist meist keine hohe Sicherheit erforderlich, da keine personenbezogenen Daten oder Währungen auf dem Chip gespeichert werden und die Ticketpreise so gering sind, dass ein Angriff meist teurer wäre als das Ticket selbst.[16] [17]

[14] Vgl. Transport for London, https://tfl.gov.uk/info-for/media/?cid=pp017, zuletzt geprüft am 06.10.2017

[15] Vgl. Marcio Almeida, https://www.blackhat.com/docs/sp-14/materials/arsenal/sp-14-Almeida-Hacking-MIFARE-Classic-Cards-Slides.pdf zuletzt geprüft am 17.10.2017

[16] Vgl. Semiconductors, 2009

[17] Vgl. Jonas Groß 2012, S. 21

Die Weiterentwicklung ist die MIFARE-Ultralight-C-Karte. Diese ist mit einer 3DES-Hardware-Verschlüsselung und einem EEPROM-Speicher mit einer Größe von 144 Byte ausgestattet. Somit ist diese Smartcard sicherer als ihr Vorgänger und trotzdem noch sehr kostengünstig.

Ein weiteres Produkt der MIFARE-Familie sind die MIFARE-PLUS-Smartcards. Diese Smartcards gibt es in mehreren Versionen, die sich grundlegend nur darin unterscheiden, was die Sicherheit betrifft. Die beiden Hauptvarianten sind mit einem EEPROM-Speicher mit einer Speichergröße von 2 oder 4 Kilobyte ausgestattet. Erstmals besteht bei diesen Smartcards die Möglichkeit, die Datenübertragung mit dem AES-Algorithmus zu verschlüsseln. Diese Abkürzung steht für „Advanced Encryption Standard“ und gilt bis zum heutigen Tag als sehr sicher eingestuftes symmetrisches Verschlüsselungsverfahren. [18] Weitere Versionen sind die MIFARE-Plus-S-IC und die Plus-X-Version. Die Plus-S-Variante bietet nur eingeschränkte Funktionen und ist als eine abgespeckte Version zu verstehen. Die Plus-X bietet alle Sicherheitsmechanismen. Das Besondere an dieser Smartcard ist außerdem der Entfernungscheck. Damit wird die Reichweite des Kommunikationspartners gemessen und soll so vor Relay-Attacken schützen. Diese versuchen die Reichweite zu erhöhen, um die Distanz zwischen einem Tag und dem Lesegerät zu überbrücken. Es wird praktisch ein weiterer Tag vorgetäuscht, um so die Informationen zum Angreifer zu leiten.[19] [20]

Die neuste Produktfamilie beinhaltet die MIFARE-DESFire-Smartcards. Die Neuerung schließt die DES-Unterstützung per Hardware ein. DES steht für „Data Encryption Standard“ und ist ein symmetrisches Verschlüsselungsverfahren. [21] Die erste Version der MIFARE-DESFire-Smartcard ist mit einem EEPROM-Speicher mit einer Speichergröße von 4 Kilobyte ausgestattet. Außerdem werden nur DES und TDES als Verschlüsselungsverfahren unterstützt. Die Weiterentwicklung ist die MIFARE-DESFire- EV1. Dort kann zwischen den Speichergrößen 2, 4 oder 8 Kilobyte gewählt werden. Außerdem werden als Verschlüsselungsverfahren DES, 3DES, 3KDES und AES unterstützt. (siehe Abbildung 7) Die

[18] Vgl. van Tilborg und Jajodia, S. 24

[19] Vgl. Frauenhofer SIT, http://www.rfid-basis.de/rfid-sicherheit.html, zuletzt geprüft am 17.10.2017

[20] Vgl. Jonas Groß 2012, S. 22–23

[21] Vgl. Furht 2006, S. 143–144

neuste Version ist die MIFARE DESFire-EV2. Zusätzlich zu allen Sicherheitsmechanismen und den großen Speichermöglichkeiten bietet diese Smartcard weitere Funktionen an. Der größte Vorteil dieser Karte ist die Möglichkeit, mehrere Applikationen gleichzeitig zu speichern. Weiterhin wurde die Reichweite erhöht, sodass eine Kommunikation in einem größeren Radius möglich ist. [22]

MIFARE hat außerdem einen Smartcard-Controller, den Smart-MX. Durch diesen können sogenannte Dual-Interface-Karten erstellt werden. „A dual-interface chip card has both contact and contactless interfaces. The "contactless" part means it has RFID chip that enables it to make payments through the RFID short-range radio communication. The "contact" part means it can be used with physical readers, either via a traditional magnetic stripe or with an EMV chip that lets it be dipped into an EMV reader." (Credit Card Glossary: Terms and Definitions) Das Besondere ist, dass alle erstellten Karten ISO/IEC 14443 A kompatibel sind. Es werden außerdem die Verschlüsselungsverfahren 3DES, AES und „Public Key Encryption" (PKE) unterstützt. [23]

Ein weiterer großer Smartcard-Hersteller ist die Firma Sony aus Japan. Mit der „Fecility Card" (FeliCa) hat Sony seine eigene Technologie entwickelt.

Neben dem Einsatz in verschiedenen asiatischen Zahlungs- und Zugangssystemen kommt diese Technik auch bei den Karten des Kreditkartenherstellers VISA zum Einsatz. Mit diesem VISA -Touch- System hat VISA sein eigenes NFC-Payment-System implementiert und nutzt dafür unter anderem diese Technologie. Die Standard-frequenz ist 13,56 MHz und die Übertragungsgeschwindigkeit beträgt zwischen 212 und 424 kbps. [24]

Auch die FeliCa-Card von Sony ermöglicht es, mehrere Applikationen auf einer Karte zu bedienen.

[22] Vgl. Semiconductors, 2016

[23] Jonas Groß 2012, S.13

[24] Vlg. Sony Global, https://www.sony.net/Products/felica/about/scheme.html, zuletzt geprüft am 10.10.2017

2.1.7 Sicherheitsbetrachtung Smartcards

In diesem Kapitel analysiert der Autor mögliche Schwachstellen und Angriffsszenarien auf Smartcards. Effing und Rankl beschreiben in ihrem Buch „Handbuch der Chipkarten“ folgende Klassen von Angriffen:

- Physikalische Angriffe
- Logische Angriffe
- Social-Engineering-Angriffe

Bei physikalischen Angriffen versucht der Angreifer in der Regel die Informationen aus den Mikrochips auszulesen oder gezielt zu manipulieren. Bei logischen Angriffen versucht der Angreifer auf Software-Ebene (Betriebssystem, Anwendungen) Schwachstellen auszunutzen. Social-Engineering-Angriffe setzen am schwächsten Glied der Sicherheitskette an, dem Menschen. Ein klassisches Beispiel ist die gezielte Manipulation eines Anwenders, um an seinen PIN zu gelangen und so, ohne eine technische Schwachstelle ausgenutzt zu haben, die benötigte Information zu erhalten. Mit steigender Nutzung diverser Sicherheitssysteme und besser werdenden Softwarelösungen gewinnt diese Angriffsart bei Kriminellen an Beliebtheit. Der Kosten-Nutzen-Faktor ist dort meist besser. [25] [26]

Welche Vorkehrungen werden gegen diese Risiken unternommen?Zum einen versucht man schon im Entwicklungsprozess Angriffe auf Mikrochips zu erschweren. Zum anderen werden die Geheimhaltung und der Datenschutz expliziter umgesetzt. Durch Geheimhaltung von wichtigen Entwicklungsinformationen können Angriffe erschwert werden.

Weiterhin kann der Manipulation der Hardware vorgebeugt werden. Wie in Kapitel 2.1.2 ersichtlich, enthalten neuere Prozessorkarten Sicherheitssensoren, um Manipulationen zu entdecken. Zusätzlich zu den Sensoren ist der Mikrochip mit einer Schutzschicht überzogen. Diese schützt vor physischem Zugriff. Durch Reverse Engineering versuchen Angreifer häufig auch die Struktur des Chips zurückzuverfolgen, um so Informationsflüsse besser

[25] Vgl. Rankl und Effing 2008, S. 12–15

[26] Vgl. Langer und Roland 2010, S. 68

nachvollziehen zu können. Dem wird vorgebeugt, indem die Bausteine zufällig angeordnet werden. Ebenso wird versucht, durch die zufällige Verteilung der Speicherzellen in der Adressstruktur, einen physischen Angriff zu erschweren. Dies wird sogar zur Laufzeit umgesetzt. Somit sind die Daten die ein Angreifer aus dem EEPROM- und dem ROM Speicher ausliest wertlos, solange der Angreifer den Verteilungsschlüssel nicht kennt. Diese Maßnahmen sind aber nur bedingt wirkungsvoll. Die Löschung eines EEPROM-Speichers durch UV-Licht kann so gut wie nicht verhindert werden. Durch den korrekten Einsatz von UV-Licht könnte ein Angreifer so die Daten manipulieren. Rankl und Effing beschreiben dieses Phänomen am Beispiel einer elektronischen Geldbörse. Durch den gezielten Einsatz von UV-Licht ist es Angreifern unter Umständen möglich, die Geldbörse mit dem Maximalbetrag aufzuladen. Um dem vorzubeugen, müssen schon von den Softwareentwicklern Prüfmechanismen auf Softwareebene eingebaut werden, die den Zustand der Hardware prüfen und gegebenenfalls in Entscheidungen mit einbezogen werden. [27]

2.2 RFID (Radio-Frequency Identification)

Die Abkürzung RFID steht für „Radio Frequency Identification“ und bedeutet übersetzt „Identifizierung mit elektromagnetischen Wellen“. [28]

Die RFID-Technologie dient der Kennzeichnung und Identifikation von Gegenständen, Tieren und Personen. Bestandteile eines rudimentären RFID-Systems sind ein Lesegerät und ein RFID-Tag. Die Tags sind Transponder mit Mikroprozessor und Speicher sowie integrierter Schaltung und Antenne. Die Kommunikation funktioniert kontaktlos über Radiowellen. [29]

Erstmals wurde die RFID-Technologie im Zweiten Weltkrieg in Flugzeugen und Panzern implementiert. Dies diente der sogenannten Freund-Feind-Erkennung und sollte das Erkennen

[28] Vgl. Franke und Dangelmaier 2006, S. 8

[29] Vgl. Kern 2007, S. 1

von feindlichen Fahrzeugen und Flugzeugen erleichtern. Auch heutzutage werden kontaktlose Übertragungstechnologien vom Militär eingesetzt. [30] [31]

Eine der ersten kommerziellen Entwicklungen ist die bis heute genutzte elektronische Diebstahlsicherung. Diese Anwendung ist auf die Erfindung der integrierten Schaltkreise 1958 zurückzuführen. „Die Transponder speichern 1 Bit, welches die Information codiert, ob ein Produkt bezahlt wurde oder nicht." (Langer und Roland 2010, S. 12) Seit den 1990er Jahren existieren nun kostengünstige Lösungen, um Volumenapplikationen wie Eintrittskarten, Tierkennzeichnung, Skipässe, Studentenausweise und vieles mehr zu automatisieren. Im November 2010 wurde der neue Personalausweis mit einem RFID-Chip ausgestattet. Dieser speichert persönliche Informationen, wie biometrische Daten und optional Fingerabdrücke und Lichtbild. [32]

Die RFID-Technologie ist seit Jahrzenten im Einsatz und dient als Grundlage für die später entwickelte kontaktlose „Near Field Communication" (NFC).

2.3 NFC (Near Field Communication)

Die Abkürzung NFC bedeutet übersetzt „Nahfeldkommunikation". Als Weiterentwicklung der RFID-Technologie handelt es sich auch bei der NFC um einen kontaktlosen Übertragungsstandard. Der Austausch der Daten erfolgt mit Hilfe von elektromagnetischer Induktion über Distanzen von bis zu 10cm. [33] Während es bei den klassischen RFID-Systemen immer eine aktive und passive Komponente gibt, sind NFC-Geräte in der Lage, „einerseits andere kontaktlose Chipkarten mit Energie zu versorgen und über bestehende Standard-Protokolle zu kommunizieren und andererseits auch eine kontaktlose Chipkarte zu emulieren. Die NFC-Chips integrieren beide Funktionalitäten-Lesegerät und Chipkarte." (Langer und Roland 2010, S. 7) Im Jahr 2002 entwickelte die deutsche Firma NXP Semiconductors in Zusammenarbeit

[30] Vgl. Kern 2007, S. 1–2

[31] Vgl. Langer und Roland 2010, S. 1

[32] Vgl. Kuri, https://www.heise.de/ct/artikel/Der-Internet-Ausweis-1111003.html, zuletzt geprüft am 05.09.2017

[33] Vgl. Langer und Roland 210, S.33

mit Sony die NFC-Technologie. Um NFC zu einem weltweit einheitlichen Standard zu machen, wurde 2004 von NXP, Sony und Nokia das NFC Forum gegründet.[34] Seitdem ist NFC vor allem für Mobiltelefonhersteller interessant. Das Telefon findet dabei Einsatz als Lesegerät für die kontaktlosen Chipkarten oder emuliert diese. Das erste größere kommerzielle Projekt wurde 2007 von Österreich eingeführt. In Zusammenarbeit von Mobilkom Austria AG, den Wiener Linien und der Österreichischen Bundesbahn wurden U-Bahnstationen, Bahnhöfe und Getränkeautomaten ausgestattet. Heutzutage ist NFC nicht mehr wegzudenken und Chipkarten oder Telefone fungieren als digitale Fahrkarte, Eintrittskarte oder Bezahlmedium. NFC ist außerdem ein wichtiger Bestandteil dieser Arbeit. Der Autor beleuchtet die verschiedenen NFC-Tags und dies schließt ein umfassendes Verständnis der Technologie ein. NFC-fähige Geräte können drei Operationsmodi bedienen: den Peer-to-Peer-Modus, den Reader/Writer-Modus und den Card-Emulation-Modus. Diese werden in den nächsten Kapiteln vorgestellt.[35]

2.3.1 Peer-to-Peer-Modus

Der Peer-to-Peer-Modus stellt die Vorrausetzungen bereit, um die Kommunikation zwischen zwei NFC-fähigen Geräten zu ermöglichen. Funktionen in diesem Modus sind zum Beispiel der Austausch von Dateien oder die Übertragung von Informationen. Die Norm ISO/IEC 18092 standardisiert diesen Modus. Der Standard basiert auf den Spezifikationen des „Logical Link Control Protocol“ (LLCP) des NFC-Forums. Im Peer-to-Peer-Modus kann zwischen zwei Kommunikationsmodi unterschieden werden, einem passiven und einem aktiven Modus. Auch hier spielt das in Kapitel 2.1.4 vorgestellte OSI-Schichtenmodell eine Rolle. Abbildung 7 zeigt die verschiedenen Schichten des Modells im Bezug auf den Datenaustausch beim Peer-to-Peer-Modus.

[34] Vgl. Holzinger, http://www.wcm.at/contentteller.php/news_story/near_field_communication_forum_gegruendet.html, zuletzt geprüft am 07.09.2017

[35] Vgl. NFC-Forum, https://nfc-forum.org/what-is-nfc/what-it-does/, zuletzt geprüft am 12.11.2017

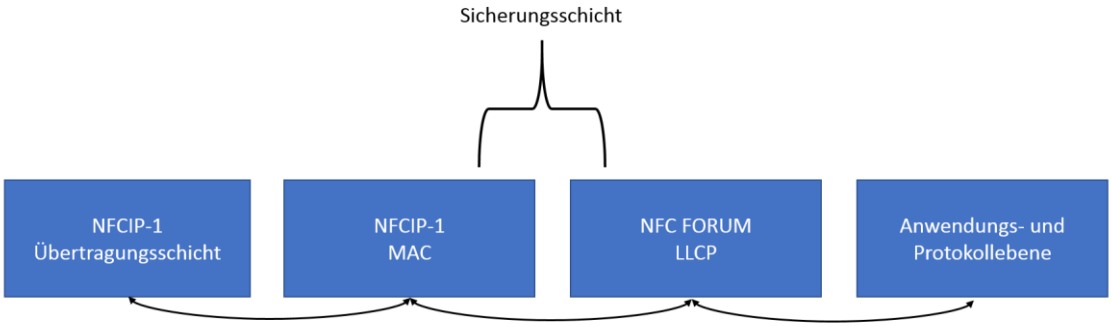

Abbildung 7: Datenaustausch im Peer-to-Peer-Modus

Die Übertragungsschicht ist nach ISO/IEC 18092 bzw. NFCIP-1 genormt. Aus diesem Grund kann sich der Initiator zwischen einem aktiven und einem passiven Kommunikationsmodus entscheiden. Die Datensicherung wird in einen „Media Access Control Layer" (MAC) und einen „Logical Link Control Layer" (LLC) aufgeteilt. Das NFC-Forum spezifiziert das LLC-Protokoll. Es definiert zwei Service-Typen: kontaktlos und kontaktbehaftet. Diese Typen sind in drei verschiedene Klassen aufgeteilt, entweder nur kontaktlos, nur kontaktbehaftet oder eine Mischform aus beiden. [36]

2.3.2 Reader/Writer-Modus

Der Reader/Writer-Modus ermöglicht NFC-fähigen Geräten, Informationen von passiven Tags auszulesen. Passive Tags sind zum Beispiel Smartposter. Die Architektur für NFC-Geräte aus dem NFC-Forum schreibt vor, dass diese Geräte nicht nur den Datenaustausch, sondern auch die Kommunikation von Geräten mit unterschiedlichen NFC-Forum-Tags ermöglichen müssen. Auf NFC-Forum-Tags wird in Kapitel 2.3.4 gesondert eingegangen. Die Bitübertragungsschichten und Verfahren zur Antikollision, die in der ISO/IEC 18092 Norm näher beschrieben werden, bilden die Basis. Nach dem Start der Kommunikation sendet der passive Transponder die Information, ob er den Peer-to-Peer-Modus unterstützt oder ob der Reader/Writer-Modus

[36] Vgl. NFC-Forum, https://nfc-forum.org/what-is-nfc/what-it-does/, zuletzt geprüft am 12.11.2017

eingesetzt werden muss. Dieser Modus bietet den Vorteil, dass die NFC-Geräte auch ältere Infrastrukturen unterstützen und in das passive Gerät integriert werden können. [37] [38]

2.3.3 Card-Emulation-Modus

Dieser Modus erlaubt es NFC-fähigen Geräten, wie Smartcards zu agieren. Das verschafft dem Nutzer die Möglichkeit, mit dem Smartphone Zahlungen zu tätigen, Tickets zu entwerten oder sich zu authentifizieren. Die Funktionen, die das NFC-fähige Gerät in diesem Modus ausführen kann, werden vom NFC-Chipsatz bestimmt. Somit ist es unter Umständen möglich, dass ein NFC-Gerät in diesem Modus mehrere kontaktlose Chipkarten emuliert.

Die Emulation kann dabei über zwei Varianten erfolgen:

- Emulation über die Software des NFC-Gerätes
- Emulation über das Secure Element

Das Secure Element wird in Kapitel 2.3.6 genauer beschrieben. Ziel des Card-Emulation-Modus könnte aber auch die Kombination mit dem Reader/Writer-Modus sein, um den Peer-to-Peer-Modus zu ersetzen. Die Anforderungen für diesen sind weitaus höher und somit bietet sich die Kombination als Alternative an. Ein weiterer Vorteil ist, dass der Card-Emulation-Modus auch mit ausgeschalteten NFC-Geräten betrieben werden kann. „Wenn zusätzlich noch der NFC-Chipsatz über das induktiv gekoppelte Lesegerät versorgt wird, lässt sich ein NFC-Gerät auch ohne eigene Stromversorgung im Card-Emulation-Modus verwenden.“ (Langer und Roland, 2010) Somit ist der Nutzer auch bei der Verwendung seines Smartphones als Smartcard nicht auf einen aufgeladenen Akku angewiesen. [39] [40]

[37] Vgl. NFC-Forum, https://nfc-forum.org/what-is-nfc/what-it-does/, zuletzt geprüft am 12.11.2017

[38] Vgl. Langer und Roland 2010, S. 99–100

[39] Vgl. Langer und Roland 2010, S. 101

[40] Vgl. NFC-Forum, https://nfc-forum.org/what-is-nfc/what-it-does/, zuletzt geprüft am 12.11.2017

2.3.4 Anwendungsgebiete

In diesem Abschnitt gibt der Autor eine Übersicht über die bekanntesten NFC-Anwendungen. Durch die wachsende Kompatibilität von der NFC-Technologie in Verbindung mit Smartphones ergeben sich immer neue Anwendungsgebiete für ein breites Nutzerspektrum, die an Benutzerfreundlichkeit kaum zu übertreffen sind. So können Bezahlvorgänge, Datenübertragungen und Zugangskontrollen in wenigen Sekunden abgewickelt werden. Dabei dienen die Smartphones, die NFC unterstützen, sowohl als passive Transponder als auch als aktive Lesegeräte.

Um dem User im Alltag schnellstmöglich zu zeigen, wo NFC genutzt werden kann, um Prozesse schneller abzuwickeln, hat das NFC-Forum die weltweit registrierte Marke „N-Mark“ eingeführt. (siehe Abbildung 8)

Abbildung 8: N-Mark (Quelle: https://nfc-forum.org/our-work/nfc-branding/n-mark/)

„Das stilisierte „N“ markiert Stellen, an denen der Benutzer durch eine kurze Berührung mit einem NFC-Gerät eine Aktion auslösen kann“. (Langer und Roland 2010)

Für Firmen, die NFC-Anwendungen bereitstellen, ist die Kennzeichnung und Verwendung der Marke kostenfrei. Es müssen allerdings die Richtlinien eingehalten werden. Das bedeutet, dass

die Tags NDEF-konform sind und es sich um NFC-Forum-Tags vom Typ1, Typ2, Typ3 oder Typ4 handelt. [41]

Die Haupteinsatzgebiete von NFC-Anwendungen sind Bezahlvorgänge, Zutrittssysteme, öffentlicher Personennahverkehr, Tickets, Eintrittskarten und Industriesysteme. In Supermärkten ist das Bezahlen mit NFC kaum noch wegzudenken und bietet eine schnellere Abwicklung an der Kasse. Rewe bietet seit Juni 2016 das kontaktlose Zahlen mit NFC in über 3000 REWE-Märkten Deutschlandweit an. Beträge unter 25 Euro können ohne die Eingabe von Pin-Codes oder der Unterschrift abgewickelt werden. [42] Auch andere Supermärkte sind mittlerweile flächendeckend mit der Technologie ausgestattet. Die größte Ursache für das hohe wirtschaftliche Interesse an der NFC-Technologie ist die Steigerung der Effizienz an Kassen und Ticketautomaten. Gerade in Großstädten wie London, Paris, Hongkong und Berlin kommen Ticketsysteme auf der Basis der NFC-Technologie zum Einsatz, um den Massenansturm von Menschen besser abwickeln zu können. London setzt dafür seit 2003 eine Smartcard als elektronische Fahrkarte ein, die auf der MIFARE-Technologie basiert. Laut eigenen Angaben werden rund 80 Prozent aller Fahrten mit der „Oyster Card" durchgeführt.[43]

Aber nicht nur für die Bezahlung der Tickets ist die NFC-Technologie im Bereich des öffentlichen Personennahverkehrs relevant. Einige Verkehrsbetriebe bieten die Möglichkeit, durch sogenannte Infopoints Fahrplaninformationen direkt auf das Smartphone zu übertragen. In Deutschland ist die wohl bekannteste Anwendung im öffentlichen Personennahverkehr das „Touch and Travel" System der Deutschen Bahn. Dieses soll dem Nutzer Geld einsparen und den Ablauf des Ticket-Systems verbessern. (siehe Abbildung 9) Bevor der Passagier den Zug betritt, muss er sein Smartphone an einen Touchpoint halten. Dasselbe gilt beim Aussteigen aus dem Verkehrsmittel. Die im Hintergrund arbeitende Software ermittelt die Route und dem Nutzer wird automatisch der beste Preis für die Strecke in Rechnung gestellt. Der Touchpoint enthält einen NFC-Tag. Dieser enthält die Informationen über den aktuellen Standort.

[41] Vgl. Langer und Roland 2010, S. 205

[42] Vgl. REWE, https://presse.rewe.de/artikel/rewe-bietet-kontaktloses-zahlen-an/, zuletzt geprüft am 10.11.2017.

[43] Vgl. Transport for London, https://tfl.gov.uk/info-for/media/?cid=pp017, zuletzt geprüft am 06.10.2017

Die Software im Hintergrund erhält also die Standortinformationen beim Einsteigen und beim Aussteigen und kann so die passende Route berechnen. [44]

Abbildung 9: Touchpoint Deutsche Bahn (Quelle: https://www.morgenpost.de/berlin-aktuell/article105042250/BVG-fuehrt-Handyticket-flaechendeckend-ein.html)

Aber auch für diverse Zutrittssysteme ist NFC eine häufig eingesetzte Technologie. Anstelle von Schlüsseln werden mittlerweile in vielen Hotels Smartcards oder Mobiltelefone eingesetzt.

[44] Vgl. Langer und Roland 2010, S. 208

2.3.5 Datenformate

Durch das NFC-Forum ist heutzutage der Datenaustausch zwischen NFC-Geräten durch eine hohe Kompatibilität gekennzeichnet. Dank Speicherung einheitlicher Datenformate ist es für Entwickler einfacher als nie zuvor, sich auszutauschen und neue Speicher- und Datenformate zu entwickeln, die konform mit den Spezifikationen des NFC-Forums sind. Der Hauptbestandteil einer NFC-Infrastruktur sind die Tags. Diese müssen gewisse Funktionen haben, um eingesetzt werden zu können. Das sind zum einen Lese- und Schreiboperationen für den Datenaustausch und zum anderen Prozesse, um den Datenspeicher vor unerlaubtem Zugriff und Manipulation zu schützen. [45] In den Spezifikationen des NFC-Forums befinden sich momentan vier Tag-Typen. (siehe Tabelle 2)

Tag-Name	Norm	Speichergröße	Beschreibung
NFC Forum Type 1	ISO/IEC 14443A	96 Byte - 2 Kilobyte	read and re-write Möglichkeit read-only
NFC Forum Type 2	ISO/IEC 14443A	48 Byte - 2 Kilobyte	Read and re-write Möglichkeit read-only
NFC Forum Type 3	(JIS) X 6319-4	0 Byte - 1 Megabyte	Entweder read oder re-write oder readonly
NFC Forum Type 4	ISO/IEC 14443	0 Byte - 32 Kilobyte	Entweder read oder re-write oderreadonly

Tabelle 2: Tag-Typen des NFC-Forum

Die NFC-Forum-Tags vom Typ 1 sind einfache Speichertags, die als Grundlage die NFC-A Technologie nutzen. NFC-A ist eine „signaling technology“. Bevor der Datenaustausch zwischen einem NFC-Lesegerät und einem Tag stattfinden kann, wird anhand der „signaling

[45] Vgl. Langer und Roland 2010, S. 109

technology“ überprüft, ob eine Kommunikation möglich ist. Dazu wird ein spezifisches Protokoll übertragen. [46]

Für den Typ 1-Tag gibt es zwei verschiedene Speichermodelle. Es ist möglich, eine statische Speicherstruktur mit einer Speichergröße von 96 Byte zu verwenden oder aber eine dynamische mit bis zu 2 Kilobyte. Bei dem genutzten Speicher handelt es sich um einen EEPROM-Speicher mit Blöcken zu je acht Bytes. EEPROM steht für „Electrically Erasable Programmable Read-only Memory“. Es handelt sich dabei um einen elektronischen Speicherbaustein, auf dem Informationen gespeichert werden. Diese können elektrisch gelöscht werden. [47] Außerdem wird noch eine Header-Rom verwendet. Dieses zeigt, ob der Transponder kompatibel ist und welche Speicherstruktur implementiert ist. [48]

Die NFC-Forum-Tags vom Typ 2 sind ebenfalls einfache Speichertags, die als Grundlage die NFC-A Technologie nutzen. Der Unterschied zum Typ 1-Tag ist, dass ein Prozess durchgeführt wird, der es ermöglicht, dass sich mehrere Tags vom Typ 2 im Wirkungskreis eines Lesegerätes befinden dürfen. Dieses Verfahren nennt man Antikollision. Auch beim Tag-Typ 2 gibt es zwei Speicherstrukturen. Zum einen die statische mit einer Speichergröße von 48 Byte und zum anderen die dynamische mit Speichergrößen bis zu 2 Kilobyte. Ein weiterer Unterschied findet sich im verwendeten EEPROM-Speicher. Bei den Tags des Typs 2 ist dieser in Blöcke zu je vier Byte aufgeteilt. [49]

Die NFC-Forum-Tags des Typs 3 sind anders als Typ 1 und Typ 2-Speichertags, die auf der NFC-F-Technologie basieren. Die NFC-F-Technologie ist ebenfalls eine „signaling technology“. Auch NFC-F verwendet ein Antikollisionsverfahren. Durch Systemcodes werden hier die Tags bestimmten Anwendungen zugeteilt. Die Speicherstruktur dieser Tags funktioniert dynamisch von 0 Byte bis zu 1 Megabyte. Der EEPROM-Speicher des NFC-Forum-Tags des Typs 3 ist außerdem in ein Dateisystem eingebettet. Somit ist es möglich, für verschiedene Servicecodes die Zugriffsrechte zu vergeben und zu ändern. Bei diesem EEPROM-Speicher

[46] Vgl. NearFieldCommunication.org,http://nearfieldcommunication.org/nfc-signaling.html, zuletzt geprüft am 20.09.2017

[47] Vgl. Mitescu und Susnea 2005, S. 107–113

[48] Vgl. Langer und Roland 2010, S. 110–114

[49] Vgl. Langer und Roland 2010, S. 114–116

bestehen die Datenblöcke für die verschiedenen Services aus einem oder mehreren Datenblöcken mit einer Größe von je 16 Byte.[50]

Die NFC-Forum-Tags des Typs 4 sind keine einfachen Speichertags. „Im Gegensatz zu einfachen Speichertags (Tag-Typ 1 und 2) handelt es sich bei Tag-Typ 4 um eine prozessorbasierte Smartcard-Technologie." (Langer und Roland 2010, S. 117) Diese sind konform zur Norm ISO/IEC 14443. Diese Norm ist international für kontaktlose Chipkarten gültig.[51] Die verwendete „signaling technology" variiert je nach verwendeter Chipkarte zwischen NFC-A und NFC-B. Auch diese Tags benutzen Antikollisionsverfahren, um mehrere Tags gleichzeitig aktivieren zu können. Wie bei Tags des Typs 3 ist die Speicherstruktur hier dynamisch und mit einer Speichergröße zwischen null Byte und 32 Kilobyte versehen.

Ein weiterer wichtiger Standard für die Kommunikation zwischen verschiedenen Geräten und Tags ist das „NFC Data Exchange Format"(NDEF). Durch diese Spezifikation wird erreicht, dass eine Unabhängigkeit zwischen Datenquelle und NFC-Anwendung herrscht. „NDEF ist ein einfaches binäres Datenformat, das anwendungsspezifische Nutzdaten kapselt." (Langer und Roland 2010, S. 120) Die Steuerung der sicheren Übertragung von Anwendungsdaten ist hingegen Sache der Übertragungsprotokollebene oder der Anwendungsebene. Anwendungsdaten und Metainformationen sind in sogenannten NDEF-Records gespeichert. Die Metainformationen beschreiben dabei die Architektur des NDEF-Records. Weiterhin enthalten sind dort Informationen zum Typ, zur Interpretation der Nutzungsdaten und Informationen zur Identifikation. Zusammengefasst werden die Records dann zur sogenannten NDEF-Message.[52] Dieses Protokoll ist auch für die zugrundeliegende Arbeit sehr wichtig. Durch die Unterstützung durch Android und die gute Dokumentation ist es für den Autor möglich gewesen, einen standardisierten Tag-Content zu erzeugen. Dabei kann außerdem ignoriert werden, dass jeder Tag-Typ unterschiedliche Hardware- und Speicherlayouts hat. Aufgrund der hohen Relevanz für die spätere Beleuchtung von NFC-Applikationen im Praxisteil, geht der Autor in diesem Abschnitt genauer auf das „NFC Data Exchange Format" (NDEF) ein. Der Aufbau eines NDEF-Records (Abb. 10) setzt sich aus einem Header und einem Bereich von benötigten

[50] Vgl. Langer und Roland 2010, S. 116–118

[51] Vgl. Iso.org, https://www.iso.org/standard/50942.html, zuletzt geprüft am 20.09.2017

[52] Vgl. Langer und Roland 2010, S. 120

Daten, dem sogenannten Payload, zusammen. Unter diesem versteht man die verwendeten Nutzdaten, die zur Kommunikation zwischen zwei Partnern verwendet werden. [53]

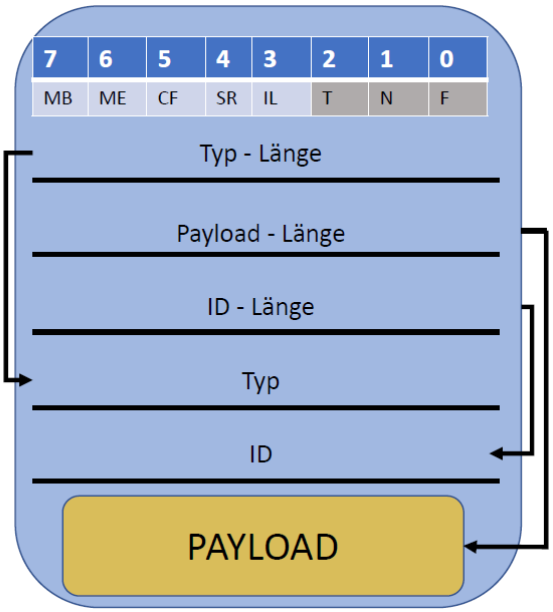

Abbildung 10: NDEF-Record

Der Header besteht aus Flags, Angaben zur Länge, Informationen über den Typ der Nutzdaten und eine ID zur eindeutigen Identifizierung. Bestandteile sind unter anderem die nun aufgelisteten Flags: MB (Message Begin), ME (Message End), CF (Chunk Flag), SR (Short Record) und IL (ID Length). Den ersten und letzten Record einer NDEF-Message zeigen die Flags MB und ME. Die Chunk Flag (CF) beschreibt die Aufteilung der Datenpakete auf die NDEF-Records. Im Falle eines „Short Record“ (SR) ist der Payload-Bereich eines NDEF-Records von 32 Bit auf 8 Bit verkürzt. IL besteht aus einem Bit und gibt an, ob der NDEF-Record Daten zur Identifikation enthält. Das Type-Name-Format (TNF) gibt an, um welches Format es sich bei dem Feld handelt. Der Bereich zur Typ-Länge ist ein 8-Bit-Wert und gibt Erkenntnis zur Länge des Type-Felds. Der Bereich zur Payload-Länge besteht im Normalfall aus 32-Bit. Sollte es sich bei dem NDEF-Record um einen „Short Record“ handeln, ist er

[53] Vgl. Martin und Weik 2001, S. 1241

allerdings nicht vorhanden. Der Bereich der ID-Länge ist 8 Bit groß und gibt Auskunft über die Länge des ID-Feldes. Das Type-Feld besteht aus einer Formatangabe, die sich aus den Vorgaben des TNF ergibt. Der Verweis auf einen anderen Record erfolgt über die ID. Der Payload-Bereich besteht aus einem Datenpaket für die Kommunikation.[54]

Einer oder mehrere NDEF-Records ergeben eine NDEF-Message. (siehe Abb. 11)

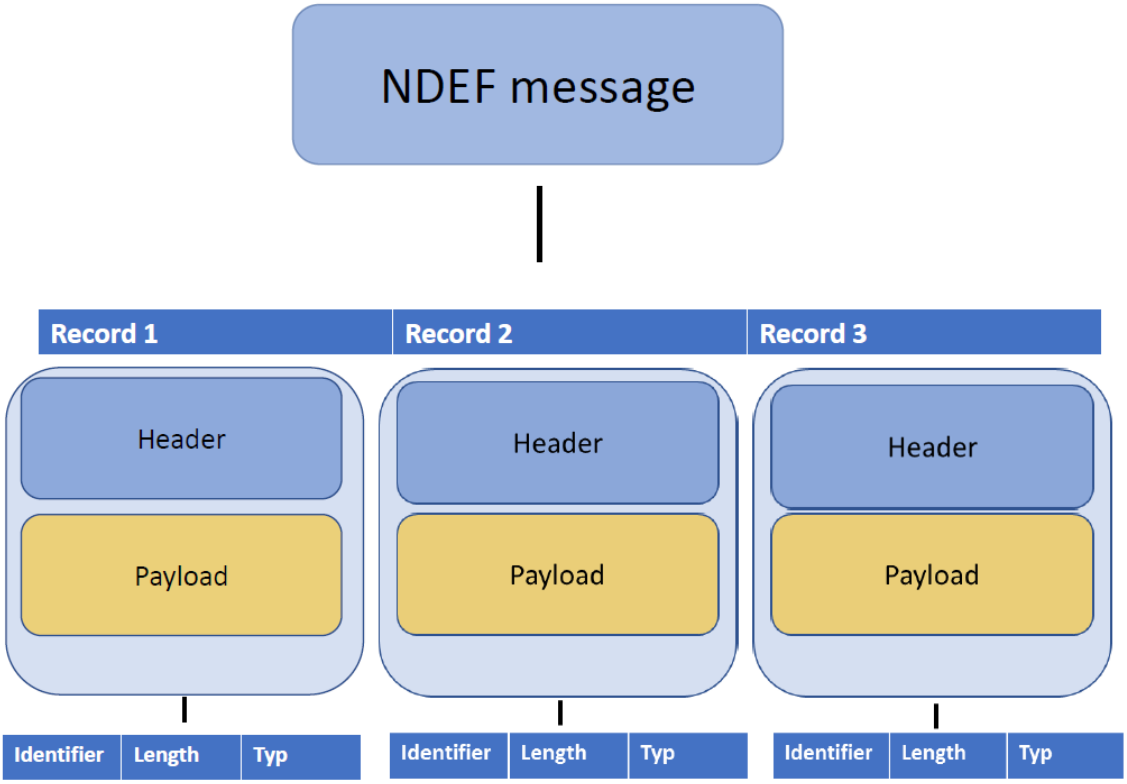

Abbildung 11: NDEF - Message

Die Flags BM (Begin Message) und EM (End Message) im Header der NDEF-Records geben das Ende und den Anfang der NDEF-Message an. Somit ergeben sich zwei Fälle. Zum einen kann eine NDEF-Message aus mehreren Records bestehen. Zum anderen gibt es einen Record,

[54] Vgl. Langer und Roland 2010, S. 120–121

in dem das BM-Bit gesetzt ist und einen, in dem das EM-Bit gesetzt ist. Sollte die NDEF-Message nur aus einem NDEF-Record bestehen, so ist in diesem Record sowohl das BM-Bit als auch das EM-Bit gesetzt.[55]

Ein weiteres Datenformat sind die MIME-Media-Types. Die Abkürzung MIME steht für „Multipurpose Internet Mail Extensions" (MIME). MIME-Media-Types verarbeiten Nachrichten zu einheitlichen Formaten. Der Einsatz dieses Datenformates dient vor allem der Spezifizierung von Textnachrichten mit beliebigen Zeichensätzen. [56] [57] Die Angabe des Types erfolgt in zwei Bereichen. Zum einen wird der Top-Level-Type festgelegt und zum anderen der Subtype. Der Top-Level-Type nimmt die erste Kategorisierung vor. Dabei unterscheidet er in „text" für Textnachrichten, „image" für Bilddateien, „video" für Videodateien, „audio" für Audiodateien und „application" für Anwendungen. Eine weitere Spezifizierung nimmt der Subtyp vor. Bei Bilddateien wird mit „image" erst festgelegt, dass es sich um eine Bilddatei handelt und danach wird das Format beschrieben, zum Beispiel „jpeg" für JPEG-Dateien. Das Ganze wird dann als MIME-Type im Format image/jpeg beschrieben. Das JPEG-Format steht für „Joint Photographic Experts Group". [58]

Durch die NDEF-Records ist die Nutzung verschiedenster Datenformate gegeben. In Verbindung mit den MIME-Media-Types können somit unterschiedliche Daten in den Records implementiert werden. „Das NDEF-Format definiert jedoch nicht, wie NFC-Geräte die Daten interpretieren und darstellen müssen. Auch MIME-Media-Types geben nur das Format der Daten an. „Bei NFC-Anwendungen stehen die Einfachheit und die herstellerübergreifende Kompatibilität im Vordergrund. Es genügt daher nicht nur, die Datenformate zu spezifizieren." (Langer und Roland 2010, S. 123)

In Betracht dessen ist es wichtig, noch ein letztes Datenformat einzubringen. Dabei handelt es sich um „NFC-Record-Type Definition" (RTD). Dieses Datenformat gibt zusätzlich zu einigen Datenstrukturen auch Verweise zum Interpretieren und Darstellen der Daten an. RTD-Spezifikationen setzen sich aus verschiedenen Basisspezifikationen und Record-Typen

[55] Vgl. Langer und Roland 2010, S. 122

[56] Vgl. Freed & Borenstein 1996, S.133

[57] Vgl. Nathaniel S. Borenstein, S.140

[58] Vgl. Langer und Roland 2010, S. 123–124

zusammen. Dabei werden in der Basisspezifikation die Rahmenbedingungen, wie zum Beispiel die Fehlerbehandlung oder die Konvertierung der Namen, von RTDs geregelt. [59]

2.3.6 NFC im Smartphone

Durch die stetig steigenden Einsatzmöglichkeiten von Smartphones steigt auch der Nutzen von NFC. Dabei wird diese Technologie eingesetzt, um Bezahlprozesse zu vereinfachen, Bluetooth- und WLAN-Verbindungen herzustellen oder Webadressen aufzurufen. In diesem Kapitel gibt der Autor einen Einblick in das Konstrukt von Organisationen, Standards und Komponenten, die durch das Bilden gemeinsamer Schnittstellen viele Möglichkeiten für die Nutzer eröffnen, NFC in den Alltag einzugliedern. [60] Um ein Smartphone mit NFC auszustatten, sind vier Hardware-Bestandteile notwendig. (siehe Abb. 12)

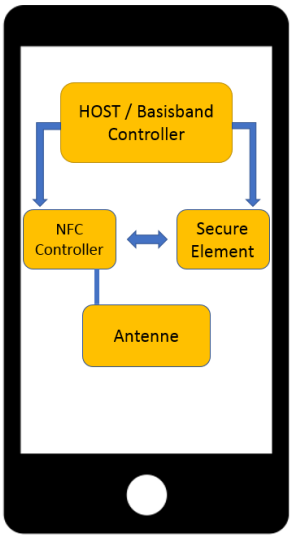

Abbildung 12: Architektur NFC - Smartphone

Die vier Hauptbestandteile eines NFC-fähigen Smartphones sind der Host-/Basisbandcontroller, das Secure-Element, der NFC-Controller und die NFC-Antenne. Der Host/Basisband-Controller oder auch „Application Execution Environment“ *(AEE)* ist für die Ausführung aller Anwendungen des Smartphones zuständig. Aufgrund der unzureichenden Sicherheitsmechanismen werden kritische Anwendungen nach Möglichkeit nicht in dieser

[59] Vgl. Langer und Roland 2010, S. 123–124

[60] Vgl. Langer und Roland 2010, S. 144–146

Ausführungsumgebung gestartet. Der NFC-Controller oder „Contactless Frontend“ ist für die Ansteuerung der NFC-Antenne zuständig. Dieser Controller verfügt über ein analoges Frontend. Weiterhin dient der Controller als Kommunikationsschnittstelle zwischen Basisband-Controller und Secure-Element. „Der NFC-Controller fungiert daher im Prinzip als Modulator und Demodulator, über den Daten über die kontaktlose NFC-Schnittstelle übertragen werden können.“ (Zefferer 2012) Um auch kritische Anwendungen in Bezug auf IT-Sicherheit ausführbar zu machen, wurde eine sichere Ausführungsumgebung, das Secure-Element oder „Trusted Execution Environment“ (TEE), eingeführt. Die Implementierung dieser Sicherheitsmechanismen ist in verschiedenen Ansätzen möglich. In den meisten Fällen ist ein Secure-Element jedoch als Hardwaremodul implementiert. Das Problem bei einer Softwarelösung wäre, dass diese in der AEE ausgeführt werden und somit erheblich an Sicherheit verlieren würde. Bei dem Einsatz von Secure-Elements als Hardwarelösungen unterscheidet man zwischen integrierten Hardware-Lösungen und externen Lösungen, wie z.B. eine Micro-SD–Karte. Die Secure-Elements sind weiterhin mit verschiedenen Betriebsarten und Komponenten kompatibel. Zwei Betriebsarten sind wesentlich und werden fast von allen Secure-Elements unterstützt. (Abb. 13)

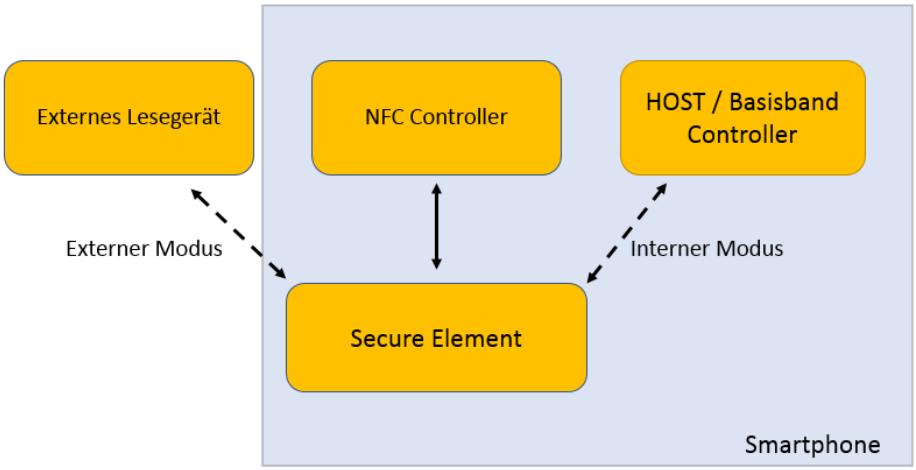

Abbildung 13: Betriebsarten Secure - Element

Bei Verwendung des Externen Modus emuliert das Secure-Element eine ICC, also eine kontaktlose Chipkarte, um einen Datenaustausch mit einem externen Lesegerät zu starten. Dabei läuft die komplette Kommunikation über den NFC-Controller. Im internen Modus wird als Lesegerät der Host/Basisband-Controller eingesetzt. In diesem Fall wird ebenfalls vom Secure-

Element eine kontaktlose Chipkarte emuliert. Die Aufgabe des NFC-Controllers bleibt bestehen. Eine Ausnahme bilden aber die Secure-Elements, die über die SIM-Karte implementiert wurden, da diese über eine direkte Verbindung zum Basisband-Controller verfügen. [61]

2.3.7 Sicherheitsbetrachtung NFC

Die Standard Protokollebenen der NFC-Technologie bieten keine Sicherheitsmechanismen, die nach derzeitigem Stand der Technik ausreichen würden. Betrachtet man diese Technologie mit dem Auge eines IT-Sicherheitsbeauftragten und vollführt eine Risikoanalyse, fallen mehrere Schwachstellen auf. Dies liegt vor allem daran, dass es sich bei der Nahfeldkommunikation um einen kontaktlosen Kommunikationskanal handelt. Diese Kanäle sind einfach manipulierbar und abhörbar. Viele Hersteller von z.B. NFC-basierten Bezahlmethoden werben mit der hohen Sicherheit. Dies stützt sich zumeist auf Fakten wie die kurze Reichweite und die schnelle Abwicklung. Die Bedrohung des unbefugten Zugriffs auf die Daten ist dennoch real. Es existieren viele Schwachstellen, die ausgenutzt werden können. Eine Angriffsform, die häufiger zum Einsatz kommt, ist die sogenannte Replay- Attacke. [62]

Im konkreten Beispiel der kontaktlosen Bezahlvorgänge könnte ein Angreifer mit einer Replay-Attacke eine Transaktion wiederholen, um sich so einen eigenen finanziellen Vorteil zu verschaffen und dies meist aus weiterer Entfernung als der Standard zulassen würde. Eine weitere Schwachstelle ist die Störung des Signals. Mithilfe eines Störsenders kann die Kommunikation komplett blockiert oder gezielt manipuliert werden. Weiterhin sind sogenannte Man-in-the-middle-Attacken (MITM) denkbar. Die MITM Attacke ist eine ältere Angriffsform die erfolgreich gegen eine Vielzahl von Kommunikationsprotokollen eingesetzt wurde. Dabei schaltet sich der Angreifer zwischen zwei Kommunikationspartner und lässt diese über sich kommunizieren. Durch diese Angriffsform kann der Angreifer die Informationen des Absenders manipulieren und so dem Empfänger falsche Informationen weitergeben. Das Risiko der meisten der eben beschriebenen Attacken kann durch kryptografische Verfahren signifikant reduziert werden. Ist die Kommunikation verschlüsselt, ist es deutlich schwerer, unautorisierten Zugriff auf die Daten zu erlangen. Aber auch sichere Verfahren zum Austausch der Schlüssel und die

[61] Vgl. Zefferer 2012, S. 7–9

[62] Vgl. van Tilborg und Jajodia 2011b, S. 1042

Herstellung eines sicheren Übertragungskanals sind nur bedingt hilfreich gegen MITM-Attacken. Auch für kryptografische Verfahren im Rahmen der NFC-Technik existieren mehrere Standards. [63] Einer der bekanntesten ist NFCP-1 Security Services and Protocol (NFC-SEC). Die Normenreihe NFC-SEC ist folgendermaßen aufgebaut. (siehe Abb. 14)

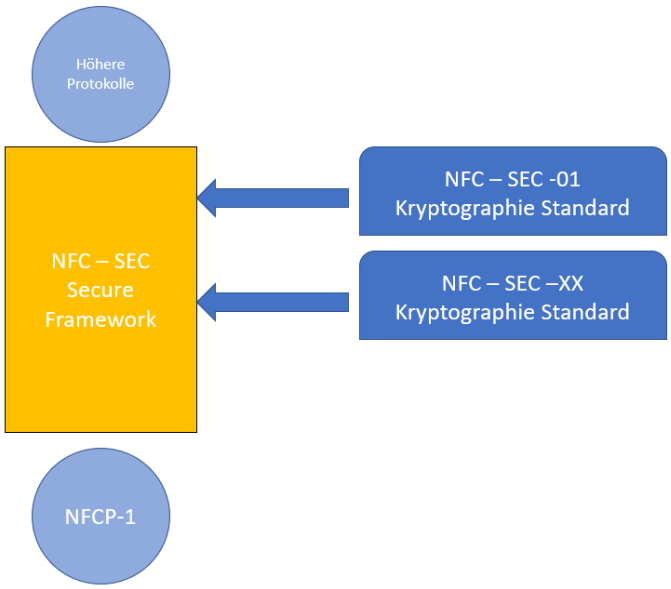

Abbildung 14: NFC-SEC

Das sichere Framework von NFC-SEC wird zwischen die verschiedenen Protokollebenen geschaltet. Die Norm umfasst Dienste und Ablaufprotokolle und setzt einen allgemeinen Rahmen für die Kommunikation. Es existieren zwei Dienste:

- Shared Secret Service (SSE)
- Secure Channel Service (SCH)

[63] Vgl. van Tilborg und Jajodia 2011b, S. 759

SSE und SCH regeln die Bedingungen für einen sicheren Schlüsselaustausch. SCH baut zusätzlich einen sicheren Übertragungskanal auf. Bestandteile des NFC-SEC-Frameworks sind mehrere Kryptographie-Standards. Der erste in der Norm beschriebene Standard ist NFC-SEC-01. Zur Verschlüsselung der Daten wird der bekannte Verschlüsselungsstandard AES (Advanced Encryption Standard) verwendet. [64] Für den Schlüsselaustausch wird das EDCH (Elliptic Curves Diffie-Hellman) Verfahren verwendet.[65] Im letzten Abschnitt beschreibt der Autor nun ein aus der Praxis bekanntes Angriffsszenario. 2015 haben die Sicherheitsforscher Jose Vila und Ricardo J. Rodriguez einen Angriff mithilfe eines Smartphone-Trojaners erfolgreich ausgeführt. Das Opfer wird dazu gebracht, eine präparierte App aus dem Internet herunterzuladen und zu installieren. Diese Applikation überwacht die Umgebung des Smartphones des Opfers auf NFC-Geräte. Sobald eine NFC-fähige Kreditkarte in der Nähe ist, erhält der Angreifer eine Nachricht auf seinem Smartphone. Dieses muss er dann an ein NFC-fähiges Lesegerät halten und kann so eine Transaktion ausführen. Das Gerät des Opfers muss dazu nicht einmal gerootet sein. [66]

[64] Vgl. van Tilborg und Jajodia 2011b, S. 24

[65] Vgl. van Tilborg und Jajodia 2011b, S. 340–345

[66] Vgl. Dennis Schirrmacher 2015

3 Herangehensweise zur Entwicklung einer Android-Applikation zum Auslesen und Beschreiben von ausgewählten NFC-Tags

3.1 Aufsetzen der Entwicklungsumgebung

Bevor mit der Entwicklung einer Applikation gestartet werden kann, muss eine Entscheidung über die Entwicklungsumgebung getroffen werden. Dort gibt es verschiedene Möglichkeiten. Da die Vorstellung aller den Rahmen dieser Arbeit sprengen würde, begrenzt sich der Autor auf die Wahl, die er für dieses Projekt getroffen hat. Als Programmiersprache wird Java vorgestellt. Sie gehört zu den objektorientierten Programmiersprachen. Die Entscheidung für diese ist insofern sinnvoll, da Java von 97 Prozent aller Unternehmensdesktops genutzt wird und eine der am besten dokumentierten und am weit verbreitetsten Programmiersprachen ist. [67] Als Entwicklungsumgebung wurde Eclipse verwendet. Es ist eine sogenannte „Integrated Development Environment“ (IDE), also eine integrierte Entwicklungsumgebung. Diese nimmt dem Nutzer viele Aufgaben ab und ermöglicht einen reibungsloseren Ablauf des Programmiervorganges.[68] [69] Die Nutzung dieses Werkzeugs ist kostenfrei. Da es in diesem Abschnitt um die Entwicklung einer Android-Applikation geht, ist ein weiterer Bestandteil von äußerst großer Relevanz. Das sogenannte „Software-Development-Kit“ (SDK). Es vereint Programme und Werkzeuge, die von einem Programmierer genutzt werden, um die Software zu entwickeln. Hierfür existiert das Android-Software-Development-Kit. Weiterhin existiert ein sehr hilfreiches Plugin für Eclipse, die sogenannten „Android-Development-Tools“ (ADT). Der erste Schritt ist also der Download und die Installation der Entwicklungsumgebung Eclipse. Danach erfolgt der Download und die Einbettung des Android SDKs sowie der Download und

[67] Vgl. Java, https://java.com/about, zuletzt geprüft am 27.09.2017

[68] Vgl. Milinkovich, http://www.eclipse.org/org/#about, zuletzt geprüft am 27.09.2017

[69] Vgl. Delgrande und Faber 2011, S.100-104

die Installation des ADT-Plugins.[70] Für weitere Informationsanleitungen und Hinweise siehe https://developer.android.com/studio/index.html.

3.2 Grundlagen für die Programmierung einer Android NFC-App

Bei der Programmierung einer Android-Applikation empfiehlt sich als Erstes die Festlegung der Rechte, die eine App benötigt, um ihren Funktionsumfang zu erfüllen. Die Erlaubnis für die Verwendung gewisser Geräte-Funktionen und Adapter bildet dabei keine Ausnahme. So muss auch eine Berechtigung für den NFC-Adapter festgelegt werden. Berechtigungen werden in der Datei „AndroidManifest.xml“ festgelegt. Jede Applikation muss dieses File im Stammverzeichnis besitzen. Es enthält essentielle Informationen über die App und muss an das System weitergeleitet werden, bevor sie genutzt werden kann. Die Hauptbestandteile des Manifest-Files sind die Deklarierung des verwendeten Java-Pakets, die Beschreibung der Prozesse, die Bereitstellung der Komponenten und Rechte, die Android-Version, die mindestens vorhanden sein muss, um die Applikation nutzen zu können, die Auflistung der Bibliotheken, mit der die App verbunden sein muss sowie die Beschreibung der Komponenten, Aktivitäten und Services. Das File ist folgendermaßen aufgebaut:

```
<?xml version="1.0" encoding="utf-8"?>

<manifest>

    <uses-permission />
    <permission />
    <permission-tree />
    <permission-group />
    <instrumentation />
    <uses-sdk />
    <uses-configuration />
    <uses-feature />
    <supports-screens />
    <compatible-screens />
    <supports-gl-texture />

    <application>

        <activity>
            <intent-filter>
                <action />
                <category />
```

[70] Vgl. Android Developer, https://developer.android.com/studio/index.html, zuletzt geprüft am 10.10.2017

```
                <data />
            </intent-filter>
            <meta-data />
        </activity>

        <activity-alias>
            <intent-filter> . . . </intent-filter>
            <meta-data />
        </activity-alias>

        <service>
            <intent-filter> . . . </intent-filter>
            <meta-data/>
        </service>

        <receiver>
            <intent-filter> . . . </intent-filter>
            <meta-data />
        </receiver>

        <provider>
            <grant-uri-permission />
            <meta-data />
            <path-permission />
        </provider>

        <uses-library />

    </application>

</manifest>
```

Die Struktur der Datei ist wichtig, um die Code-Teile der kommenden Abschnitte zu verstehen. Android verpflichtet jede App, Zugriffsrechte beim Nutzer anzufragen. Dies soll verhindern, dass eine Applikation auf alle Bestandteile des Smartphones Zugriff hat, obwohl diese für den vollen Funktionsumfang der Applikation nicht gebraucht werden. Beispiele für solche Berechtigungen sind zum Beispiel das Abfragen des Standorts oder der Zugriff auf die Kontakte. Um also die Funktion der Applikation zu gewährleisten, ist es unbedingt notwendig, Berechtigungen oder im Englischen Permissions zu integrieren. Diese Berechtigungen werden im oberen Teil der „AndroidManifest.xml“ in folgender Syntax deklariert:

<uses-permission android:name=“android.permission.NFC“/>

Ein weiterer Schritt ist die Festlegung, dass die Applikation nur auf Geräten funktioniert, die mit NFC ausgestattet sind. So kann eine Filterung im Google Play Store erreicht und die App den Nutzern angezeigt werden, deren Geräte mit NFC ausgestattet sind.

Das Google-Play-Store ist die Plattform, auf der Android-Applikationen, Bücher, Musik und vieles mehr für Android-Nutzer zur Verfügung gestellt werden. [71] Um das benötigte Feature festzulegen, muss folgender Code vor dem Applicaton-Node im „AndroidManifest.xml“ implementiert werden.

<uses-feature android:name=“android.hardware.nfc“ android:required=“true“ />

Ein weiterer Schritt ist die Festlegung der Android-Version, die mindestens benötigt wird, um die Applikation auszuführen. Bei der Auswahl der Version müssen mehrere Punkte beachtet werden. Zum einen ist es wichtig, möglichst viele Nutzer zu erreichen. Zum anderen sollte aber darauf geachtet werden, dass die Version, die mindestens vorhanden sein muss, alle Bedingungen erfüllt, um den Funktionsumfang der App in sinnvollem Rahmen ausführen zu können. Auch bezüglich der Sicherheit sollte lieber eine neuere Version verwendet werden. NFC wurde in der Android Version 2.3, API-Level 9 eingeführt. API steht für application programming interface und bezeichnet den Teil des Programmes, der zur Anbindung an das System benutzt wird. Da aber mehrere sehr wichtige Features erst in API-Level 10 eingeführt wurden, ist diese Versionierung als minimalste Anforderung zu empfehlen.

Auch das wird im AndroidManifest.xml wie folgt festgelegt:

<uses-sdk android:minSdkVersion=“10“ />

Zu den ersten Schritten gehört weiterhin die Implementierung des Codes, um zu testen, ob ein Gerät von der Hardware her in der Lage ist, NFC zu nutzen beziehungsweise, ob es einen NFC-Adapter besitzt. Android stellt einen einfachen Weg bereit, um Adapter-Objekte anzufragen:

```
Adapter nfcAdapter = NfcAdapter.getdefaultAdapter(this);
if (nfcAdapter != null && nfcAdapter.isEnabled()) {
Toast.makeText (this, "NFC ist verfügbar.", Toast.LENGTH_LONG).
show();
}else {
Toast.makeText (this. "NFC ist auf diesem Gerät nicht verfügbar.",
Toast.LENGTH_LONG) .show();
```

[71] Vgl. McIlroy 2016, S. 1346–1370

Als Erstes wird die Methode getdefaultAdapter([context]) aufgerufen und überprüft, ob eine valide NFC-Adapter - Instanz gemeldet wird. Wird die Instanz zurückgegeben, ist ein NFC-Adapter vorhanden. Weiterhin wird mit der Methode isEnabled() überprüft, ob die Funktion auch aktiviert ist. Wenn ja, wird der Text „NFC ist verfügbar“ ausgegeben. Sollte die Funktion deaktiviert sein, wird der Nutzer mit dem Text „NFC ist auf diesem Gerät nicht verfügbar“ darauf hingewiesen, dass die Applikation möglicherweise nicht korrekt funktioniert. [72]

3.3 Erkennung eines Tags

In den vorangegangenen Kapiteln hat der Autor genau erklärt, was ein Tag genau ist und wie dieser aufgebaut ist. Nun gilt es noch, den Nutzen von NFC-Tags zu erläutern. Diese können mit kleinen Datenmengen ausgestattet werden. So ist es möglich, Daten, wie zum Beispiel Texte, Telefonnummern, URLs oder Kontaktdaten, darauf zu speichern. Diese können, insofern eine Kommunikationsschnittstelle zwischen Tag und Gerät geschaffen wurde, von jedem NFC-fähigen Gerät ausgelesen werden. Wie der Name der Technologie Nahfeldkommunikation schon vermuten lässt, muss sich das Gerät in unmittelbarer Nähe zum Tag befinden, um dies zu erreichen. Tags fungieren im Normalfall als passives Element und speichern lediglich die Informationen. Das NFC-fähige Gerät dient als aktives Element und kann den Tag sowohl neu beschreiben als auch auslesen. Die meisten Tags sind so aufgebaut, dass sie sich für den Zeitraum der Kommunikation bei der Energiequelle des aktiven Elements bedienen und so die Kommunikation zu Stande kommt. Das alles passiert in nicht einmal einer Sekunde. [73]

3.3.1 Das Foreground-Dispatch-System

Angenommen mit dem Smartphone wird eine PDF- Datei geöffnet und es sind mehrere Applikationen installiert, die dafür in Frage kommen: Für welche entscheidet sich das Smartphone? Eine Möglichkeit ist, dass der Nutzer mit der Auswahl an Möglichkeiten konfrontiert wird und eine Applikation auswählen muss. Für NFC-Applikationen ist das nicht anders als das eben beschriebene Beispiel. Das Foreground-Dispatch-System erlaubt es, einen NFC-

[72] Vgl. Subtil 2014, S. 10–12

[73] Vgl. Subtil 2014, S. 28

Intent (TAG oder P2P Event) abzufangen beziehungsweise eine Applikation zu priorisieren. Ein Intent ist ein Messaging-Object und beschreibt eine Operation, die normalerweise ausgeführt wird, um Aktivitäten zu starten. [74]

Wenn das Foreground-Dispatch-System aktiviert ist, wird der NFC-Tag direkt zu unserer Applikation weitergeleitet und der Nutzer muss keine Auswahl treffen, selbst wenn mehrere Applikationen installiert sind, die für die Aufgabe in Frage kommen würden. Die Aktivierung des Foreground Dispatch Systems kann mit folgender Methode realisiert werden:

```
public void enableForegroundDispatch() {
            Intent intent = new Intent(activity, activity.getClass()).add-
Flags(Intent.FLAG_ACTIVITY_SINGLE_TOP);
            PendingIntent pendingIntent = PendingIntent.getActivity(activ-
ity, 0, intent, 0);
            IntentFilter[] intentFilter = new IntentFilter[] {};
            String[][] techList = new String[][] { { an-
droid.nfc.tech.Ndef.class.getName() }, { android.nfc.tech.NdefFor-
matable.class.getName() } };
            if (Build.DEVICE.matches(".*generic.*")) {
// clean up the tech filter
                  techList = null;
            }
            nfcAdapter.enableForegroundDispatch(activity, pendingIntent, in-
tentFilter, techList);
      }
```

Quelle: Subtil 2014, S.44-45

Die Deaktivierung des Foreground-Dispatch-Systems kann mit folgender Methode realisiert werden:

```
public void disableForegroundDispatch() {
      nfcAdapter.disableForegroundDispatch(activity);
}
```

[74] Vgl. Subtil 2014, S. 28–29

Wie kann dieses System in den Gesamtkontext einer Applikation eingebettet werden und wie funktioniert es?

In die Main-Activity der Applikation ist die Methode onResume() implementiert:

```
protected void onResume() {
	super.onResume();
	if (nfcHelper.isNfcEnabledDevice()) {
		nfcHelper.enableForegroundDispatch();
	}
}
```

Wenn der Nutzer die App startet und die Main-Activity ausgeführt wird, wird die onResume() Methode gestartet. Dadurch wird das Foreground-Dispatch-System auf dem NFC-Adapter aktiviert und jeder eingehende Intent wird an die Main-Activity weitergeleitet. Für neu eingehende Intents kann eine Methode onNewIntent() implementiert werden:

```
protected void onNewIntent (Intent intent) {
	Toast.makeText(this, "NFC intent obtained", TOAST.LENGTH_SHORT)
	.show();
	super.onNewIntent(intent);
}
```

In der Praxis funktioniert dieses Zusammenspiel sehr gut. Eine zusätzliche Methode onPause() wird aufgerufen, wenn das Gerät des Users in den sleep mode geht. Das bedeutet, das Foreground-Dispatch-System wird wieder deaktiviert. Sollte der User nun wieder die Applikation öffnen, wird die onResume() Methode aufgerufen und das System wieder aktiviert.

3.3.2 NFC Data Exchange Format (NDEF)

Es wurde ein Standard mit dem Namen „NFC Data Exchange Format“ (NDEF) eingeführt, der es ermöglicht, zwischen verschiedenen Geräten und mit verschiedenen Tags zu kommunizieren. „NDEF is a lightweight, binary, message format that can be used to encapsulate several custom payloads.“ (NFC Data Exchange Format (NDEF), NFC Forum, 2006). Durch das Nutzen dieses Standards ist es möglich, einen standardisierten Tag-Content zu erzeugen, welcher von den meisten Tag-Typen genutzt wird. Dabei wird außerdem ignoriert, dass jeder Tag-Typ unterschiedliche Hardware- und Speicher-Layouts hat.
Eine NDEF-Message setzt sich aus folgenden Bestandteilen zusammen: (siehe Abbildung 15)

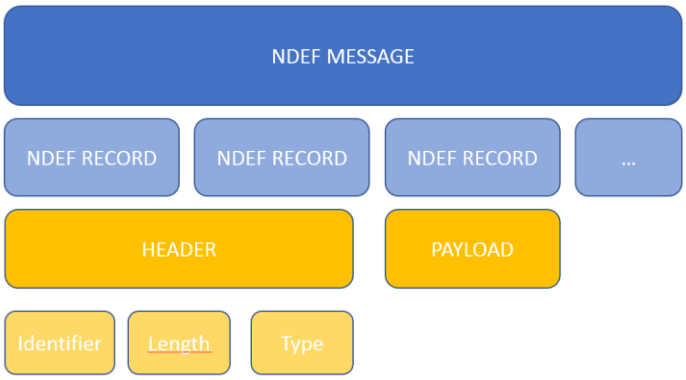

Abbildung 15: Bestandteile einer NDEF-Message

Sie kann aus einem oder mehreren Records bestehen und jeder Record hat einen eigenen Header und einen eigenen Payload. Im Header sind die Informationen zur Länge, zum Typ und zur Identifizierung gespeichert. Die NDEF-Records besitzen verschiedene Eigenschaften. Die vier Haupteigenschaften sind das „Type Name Format“ (TNF), die „Record Type Definition“ (RTD), die ID, und der Payload. TNF definiert, wie der Typ-Bereich interpretiert werden soll. RTD hilft Android im Zusammenspiel mit TNF, um die korrekte NDEF-Message zu erstellen und den richtigen Intent anzusteuern. Die ID überlässt es dem Nutzer, eine eindeutige Identifizierung für den Record vorzunehmen. Der Payload enthält Daten, die auf den Record transportiert werden.

Android unterstützt dieses Protokoll und somit ist es gut dokumentiert und erleichtert Entwicklern vieles. Für Applikationen bedeutet das eine sehr gute Filterung der Tags. In der vom Autor beschriebenen Methodik bezüglich des Foreground-Dispatch-Systems ist folgende Implementierung in die Methode enableForegroundDispatch() möglich, um nach diesem Format zu filtern.

Android stellt zwei Klassen für die Arbeit mit NDEF bereit:

- Android.nfc.tech.Ndef: Diese Klasse enthält Methoden zum Abrufen und Verändern von NdefMessage Objekten auf einem Tag.
- Android.nfc.tech.NdefFormatable: Diese Klasse enthält Methoden zum Formatieren von kompatiblen Tags. [75]

3.4 Beschreiben eines Tags

Die Klasse Ndef stellt eine Methode get(Tag tag) bereit, die es ermöglicht, eine Instanz derselben Klasse beziehungsweise einen Tag zurückzugeben. Im Anschluss daran muss eine Verbindung zum Tag hergestellt werden. Dazu wird die Methode connect() beschrieben. Wenn die Verbindung hergestellt wird, kann mit der Methode writeNdefMessage(NdefMessage msg) der Tag beschrieben werden.[76]

Die Filterung der eingehenden Tags mit Hilfe des Foreground-Dispatch-Systems kann folgendermaßen umgesetzt werden:

```
private void enableForegroundDispatch() {
        Intent intent = new Intent(this, MainActivity.class).addFlags(In-
tent.FLAG_RECEIVER_REPLACE_PENDING);
        PendingIntent pendingIntent = PendingIntent.getActivity(this, 0, intent, 0);

        IntentFilter[] intentFilter = new IntentFilter[] {};

        String[][] techList = new String[][] { { android.nfc.tech.Ndef.class.getName() }, {
android.nfc.tech.NdefFormatable.class.getName() } };

        if (Build.DEVICE.matches(".*generic.*")) {
                // clean up the tech filter when in emulator since it doesn't work
                // properly.
                techList = null;
        }

        nfcAdapter.enableForegroundDispatch(this, pendingIntent, intentFilter, techList);
```

[75] Vgl. Subtil 2014, S. 43–50

[76] Ndef, https://developer.android.com/reference/android/nfc/tech/Ndef.html, zuletzt geprüft am 16.10.2017

```
}
```

Quelle: Subtil 2014, S. 45

Mit der Methode writeNdefMessage(NdefMessage msg) kann der Tag dann beschrieben werden.

```
private boolean writeNdefMessage(Tag tag, NdefMessage ndefMessage) {

        try {
                if (tag != null) {
                        Ndef ndef = Ndef.get(tag);
                        if (ndef == null) {
                                return formatTag(tag, ndefMessage);
                        } else {
                                ndef.connect();
                                if (ndef.isWritable()) {
                                        ndef.writeNdefMessage(ndefMessage);
                                        ndef.close();
                                        return true;
                                }
                                ndef.close();
                        }
                }
        } catch (Exception e) {
                Log.e("formatTag", e.getMessage());
        }
        return false;
}
```

Quelle: Subtil 2014, S. 46

Sollte beim Aufrufen der Methode get (Tag tag) nichts zurückgegeben werden, liegt es daran das der Tag nicht korrekt formatiert wurde. Eine korrekte Formatierung kann mit der Methode formatTag() vorgenommen werden:

```
private boolean formatTag(Tag tag, NdefMessage ndefMessage) {
        try {
                NdefFormatable ndefFormat = NdefFormatable.get(tag);
                if (ndefFormat != null) {
                        ndefFormat.connect();
                        ndefFormat.format(ndefMessage);
                        ndefFormat.close();
                        return true;
                }
        } catch (Exception e) {
                Log.e("formatTag", e.getMessage());
        }
        return false;
}
```

Quelle: Subtil 2014, S. 46

Nach erfolgreicher Beschreibung muss nur noch die Methode close() aufgerufen und die Verbindung somit beendet werden. In der Praxis müssen die Methoden nun nach Bedarf an die verschiedenen Formate angepasst werden. Auch die Beschreibung der Tags mit externen Datentypen ist möglich. Der am meisten verbreitete Content auf NFC-Tags ist der „Uniform Ressource Identifier“ (URI). URI ist dazu da, um Webseiten, Online-Services oder einen Link zu kommunizieren. Diese sind so beliebt, da es sich um kleine Datenmengen handelt, die zu großen Datenmengen beziehungsweise informationsreicheren Ressourcen führen. Außerdem sind URI formatierte Ndef- Records am einfachsten zu erstellen. Ein weiteres interessantes Format sind die „Multipurpose Internet Mail Extensions“ (MIME). Diese Records bieten die Möglichkeit, (insofern der Tag genügend Speicherplatz hat) HTML Pages, Dokumente, Bilder und vieles mehr zu speichern. Der Beschreibung von Tags mit spezifischem Content sind keine Grenzen gesetzt und das NDEF - Format bietet Entwicklern die Möglichkeit, großen Kompatibilitätsproblemen aus dem Weg zu gehen. Dank dieser Hilfestellungen und der gut ausgeprägten Community, ist es möglich, NFC-Tags mit verschiedensten Größen, Strukturen und Formaten einzuführen. [77]

[77] Vgl. Subtil 2014, S. 62–84

3.5 Auslesen eines Tags

Auch beim Auslesen von NFC-Tags ist es sinnvoll das NDEF-Format zu nutzen.

„The Android system uses intents to carry data between activities, applications, and events. The same logic is applied in NFC." (Subtil 2014, S. 99)

Die erste Aufgabe besteht darin, die NDEF-Message aus dem Tag zu extrahieren. Der Autor Vitor Subtil stellt beim Kauf seines Buchs „Near Field Communication with Android Cookbook" eine sehr gut dokumentierte NFC-Hilfs-Klasse bereit. Die Methode getNdefMessageFromTag() ist exzellent zum Auslesen der NDEF-Message geeignet:

```
public NdefMessage getNdefMessageFromTag(Tag tag) {
        NdefMessage ndefMessage = null;
        Ndef ndef = Ndef.get(tag);
        if (ndef != null) {
                ndefMessage = ndef.getCachedNdefMessage();
        }
        return ndefMessage;
}
```

Quelle: Subtil 2014, S. 102

Die eigentlichen Daten sind allerdings in den NDEF-Records gespeichert. Die NFC-Hilfs-Klasse hat auch hier eine Methode bereitgestellt, um den ersten NDEF-Record auszulesen. Die Syntax für weitere Records, falls vorhanden, muss nur noch angepasst werden.

```
public NdefRecord getFirstNdefRecord (NdefMessage ndefMessage) {
        NdefRecord ndefRecord = null;
        NdefRecord[] ndefRecords = ndefMessage.getRecords ();
        if (ndefRecords != null && ndefRecords.length >0) {
                ndefRecord = ndefRecords [0];}
        return ndefRecord;
}
```

Quelle: Subtil 2014, S. 103

Android-Systeme speichern den Tag-Typ in dem ersten NDEF-Record. Somit ist das auch in den meisten Fällen der Record, auf dem die gewünschten Daten gespeichert sind. Eine weitere Möglichkeit ist mit einer Methode getRecords() alle Records auszugeben und nach den Datentypen zu testen. Um an die Rohdaten zu gelangen, empfiehlt sich die getPayload() Methode der NDEF-Message-Instanz. Diese Methode gibt die Daten in Byte zurück. Nun kommt es auf den TNF und den RTD an, welche Repräsentation der Daten am sinnvollsten ist sowie auf den genauen Datentyp, der ausgelesen werden soll. Um einen Tag mit Text zu beschreiben, muss ein Byte-Array erstellt werden, welches das Status-Bit enthält, was anzeigt, um welche Zeichenkodierung, Language-Code und String-Typ es sich handelt. Um nun den String auszulesen, ist die umgekehrte Vorgehensweise notwendig. Das Byte-Array muss rückwärts aufgeschlüsselt werden. Für mit Text formatierte Tags hält die NFC-Hilfsklasse die Methode getTextfromNdefRecord() bereit. [78]

```
public String getTextFromNdefRecord(NdefRecord ndefRecord) {
        String tagContent = null;
        try {
                byte[] payload = ndefRecord.getPayload();
                String textEncoding = ((payload[0] & 128) == 0) ? "UTF-8" : "UTF-16";
                int languageSize = payload[0] & 0063;
                tagContent = new String(payload, languageSize + 1, payload.length - languageSize
- 1, textEncoding);
        } catch (UnsupportedEncodingException e) {
                Log.e("getTextFromNdefRecord", e.getMessage(), e);
        }
        return tagContent;
}
```

Quelle: Subtil 2014, S. 106–107

[78] Vgl. Subtil 2014, S. 105–109

4 Analyse und Konzeption

4.1 Forschungsbedarf

Ein Großteil der wissenschaftlichen Arbeiten beschäftigt sich mit dem Potential von NFC-Anwendungen in der Zukunft. Dabei wird zumeist ausschließlich auf die Benutzerfreundlichkeit und die hohe Skalierbarkeit eingegangen. [79] [80] [81] Betrachtungen zum Auslesen und Beschreiben der NFC-Chips und der damit verbundenen Sicherheitsrisiken sind nicht im ausreichenden Maße verfügbar. Jonas Gross hat sich schon 2011 in seiner Diplomarbeit mit den Faktoren beschäftigt, die das Auslesen erschweren können. [82] 2013 hat das Institute of Electrical and Electronics Engineers (IEEE) eine Arbeit zu Sicherheitsrisiken und Schwachstellen von NFC-Anwendungen im Gesundheitsbereich veröffentlicht. Auch diese Veröffentlichung setzt schon einen Schritt tiefer an. Ein Buch was in vielen Arbeiten und Büchern zitiert wird ist das Handbuch der Chipkarten, welches 2008 von W. Rankl und W. Effing veröffentlicht wurde. [83] Dieses Buch beschäftigt sich ausführlich mit der Technologie und potentiellen Angriffsmethoden. Die zugrundeliegende Arbeit dient als Schnittstelle zwischen Veröffentlichungen zur skalierbaren, zukunftsträchtigen NFC-Technologie und Arbeiten zu aufwendigen Schwachstellenanalysen und Angriffsmethodiken und beschäftigt sich mit der Basis die notwendig ist um diese auszuführen. Die Forschungen die während der Recherche zu dieser Arbeit gefunden wurden, beschäftigen sich mit physischen Versuchsaufbauten. Virtualisierungslösungen und Tests in virtuellen Umgebungen konnten nicht ausfindig gemacht werden. Die zugrundeliegende Arbeit nutzt ausschließlich virtuelle Geräte und Smartcards für die Forschung.

[79] Vgl. Linnhoff-Popien . 2015, S. 379–388

[80] Vgl. Egger und Jooss 2010, S. 43

[81] Vgl. Schiereck und Tielmann 2013, S. 168-172

[82] Vgl. Jonas Groß 2012, S. 39

[83] Vgl. Rankl und Effing 2008, S. 22

4.2 Forschungsansatz

Jedes Forschungsvorhaben benötigt einen maßgeschneiderten Forschungsplan. Das Vorhaben des Autors stützt eine Beobachtung aus der Praxis. In der vorliegenden Arbeit soll dieses Phänomen wissenschaftlich greifbar gemacht werden. Im Rahmen der Planung hat sich der Autor für ein qualitatives Vorgehen entschieden. Durch die methodische Offenheit und dem flexiblen und induktiven Vorgehen kann das Forschungsziel am besten erreicht werden. Der Entscheidung für ein qualitatives Forschungsdesign folgt das Ziel, den Untersuchungsgegenstand möglichst explorativ zu ergründen und zu verstehen. Dabei entstehen theoretische Schlussfolgerungen, die als Grundlage für weitere Forschungsprojekte genutzt werden können. Weiterhin helfen die Ergebnisse dieser Forschung dem Nutzer bei der Abwägung von Risiken und dem Verständnis der zugrundeliegenden Technologien. Eine quantitative Herangehensweise ist für den Rahmen dieser Arbeit nicht sinnvoll. Zum einen liegen die Ergebnisse nicht in repräsentativer Form vor und zum anderen war ein Wissensaustausch mit Unternehmen, die sich mit der Thematik befassen, nicht möglich. Aufgrund der hohen Sicherheitsstufen dieser Firmen konnte kein Detailwissen erlangt werden.

4.3 Ableitung der Forschungsfragen

Nachdem der methodische Untersuchungsrahmen gespannt wurde, ergeben sich Forschungsfragen die es bei der Auswertung zu beantworten gilt. Die übergeordnete Forschungsfrage ist:

FF1: Welche Voraussetzungen sind nötig, um Smartcards mit NFC-Chips auszulesen oder zu beschreiben?

FF1.1: Wie viel technisches Knowhow wird benötigt, um Smartcards mit NFC-Chips auszulesen oder zu beschreiben?

FF1.2: Wie viel zeitlicher und finanzieller Aufwand ist nötig, um Smartcards mit NFC-Chips auszulesen oder zu beschreiben?

FF1.3: Existieren Smartcards mit NFC-Chips die schwieriger auszulesen sind als andere?

4.4 Methodisches Vorgehen

4.4.1 Aufsetzen der Testumgebung

Bei der Entwicklung von Android Applikationen empfiehlt es sich, zum Testen, virtuelle Geräte zu nutzen. Das hat den Vorteil, die App für verschiedenste Android-Versionen testen zu können. Weiterhin bieten virtuelle Geräte eine große Anzahl von Adaptern. Für unsere Applikation benötigen wir ein Gerät mit NFC-Adapter. Die Macher von Open NFC stellen auf Ihrer Website verschiedene virtuelle Geräte zur Verfügung www.open-nfc.org). Für mein Projekt habe ich die Open NFC Android Edition genutzt, welche es mir möglich macht, Android Geräte mit NFC Funktion zu simulieren. Diese sind nutzbar wie physische Geräte. Zu den AVDs (Android Virtual Devices) stellt Open-NFC die passenden SDKs (Software Development Kits) bereit. Diese beinhalten eine Software, welche die Kommunikation zwischen Simulator und AVD ermöglicht, das sogenannte Connection-Center. Weiterhin beinhaltet das SDK den sogenannten NFC-Simulator. Dieser erlaubt die Simulation verschiedener Tags und die Überprüfung der Funktionsweise des virtuellen Gerätes. Durch die integrierte Eclipse-Bridge ist es außerdem möglich die Applikation direkt auf dem AVD zu installieren. Leider ist es nun so dass nach Abschluss dieser praktischen Arbeit die Website www.open-nfc.org vom Netz genommen wurde. Für zukünftige Experimente muss also anderweitig Abhilfe geschaffen werden. [84]

Um die beschriebene Testumgebung aufsetzen zu können müssen folgende Schritte ausgeführt werden.

1. Navigation auf die Webseite http://open-nfc.org

2. Navigation auf den Reiter „Downloads“

3. Aus der Downloadliste *4.4.1 Open NFC for Android release* auswählen.

[84] Vgl. Subtil 2014, S. 13–17

4. Erstellen eines Ordners und Extraktion des Archivs

5. Im extrahierten Archiv sollte sich nun ein Ordner mit dem Namen android_sdk und ein Ordner mit dem Namen OpenNFC_Addon befinden. Dort ist ein Android-Image enthalten, auf welchem ein AVD (Android Virtual Device) erzeugt werden kann.

Da die Webseite http://open-nfc.org nach Initialisierung dieser Arbeit den Support für die beschriebene Testumgebung aufgegeben hat und die Webseite vom Netzt genommen wurde, müssen für zukünftige Experimente andere Lösungen gefunden werden. Alle Downloadarchive liegen dieser Arbeit bei. [85]

Im nächsten Schritt muss die *Open NFC SDK Edition* installiert werden. Dies ermöglicht uns, mit dem virtuellen Gerät zu kommunizieren, verschiedene Tags zu simulieren und den Datenaustausch zu initiieren. Folgende Schritte sind für die Installation notwendig.

1. Das Archiv *4.4.1 SDK release* in den in der vorherigen Betriebsanweisung erzeugten Ordner extrahieren.

2. In diesem Archiv befindet sich ein Ordner mit dem Namen *connection_center* und ein Ordner mit dem Namen *nfc_simulator.*

Als nächstes muss das *Open NFC Android Edition Add-On* in das existierende *Android SDK* implementiert werden. In der folgenden Anweisung beschreibt der Autor die Implementierung des *Open NFC Plugins* in das *Android SDK*.

1. Navigation zum erzeugten Ordner und zum Verzeichnis *Open NFC SDK for Android v4.4.1 (13751)*

2. Einleitung Kopiervorgang des Ordners *addon_open_nfc_4.4.1_android_4.0.3.*

3. Navigation zum *Android SDK* Installationsverzeichnis

[85] Vgl. Subtil 2014, S. 15–17

4. Einfügen des in Schritt zwei kopierten Ordners. Die Frage ob beide Ordner zusammengefügt werden sollen ist mit JA zu beantworten.

5. Starten des Android SDK Managers. Unter Windows kann dieser mit der beiliegenden SDK Manager.exe installiert werden. Danach ist das OPEN NFC-Addon unter der gewählten Android-Version ersichtlich. (siehe Abb. 16)

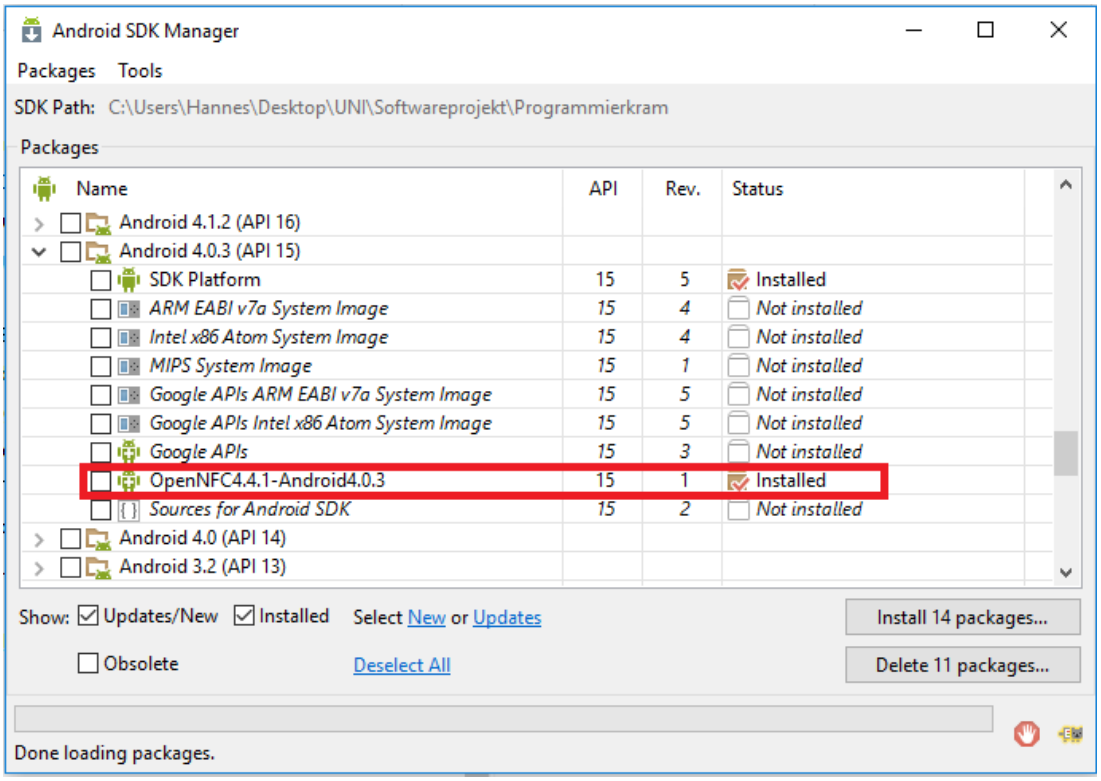

Abbildung 16: Android SDK Manager

Im nächsten Schritt muss ein AVD mit aktiviertem NFC-Adapter erzeugt und konfiguriet werden. Dies gelingt mit der Ausführung folgender Schritte:

1. Navigation zum Ordner *Android SDK*

2. Starten des *SDK Managers* und Navigation auf *Tools/Manage AVDs* (siehe Abb. 17)

3. Bei Klick auf Manage AVDs... öffnet sich der Android Virtual Device Manager, dort erzeugen wir ein neues virtuelles Gerät. Der Autor nutzt für sein Testgerät folgende Parameter. (siehe Abbildung 18)

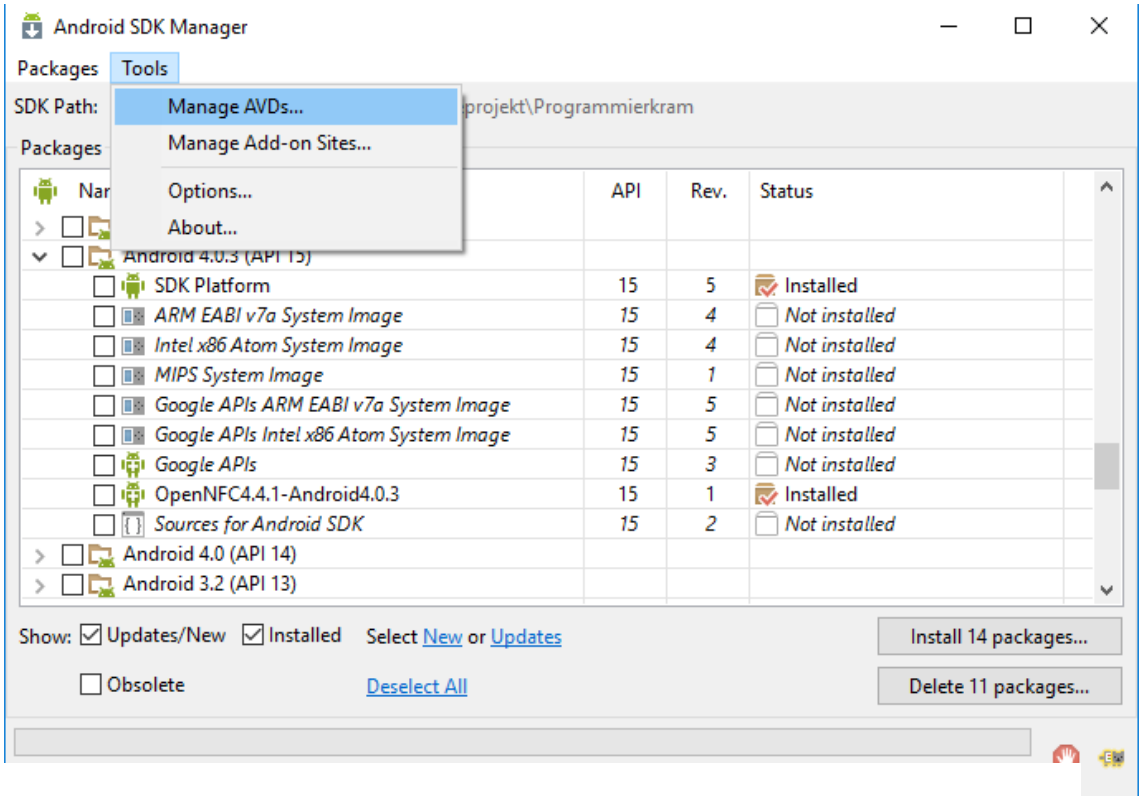

Abbildung 17: Manage AVDs

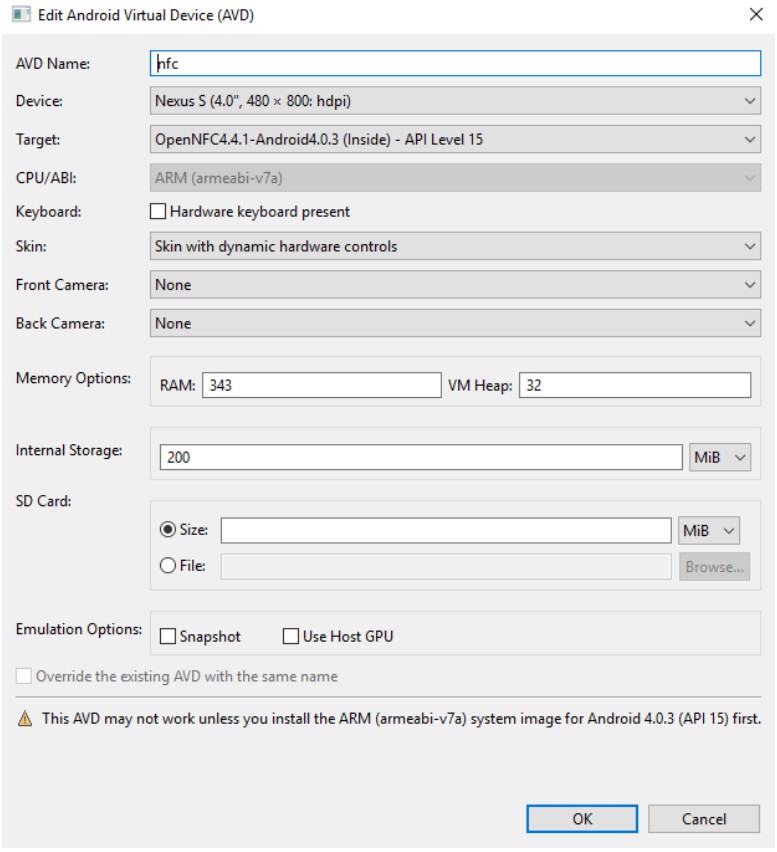

Abbildung 18: Parameter AVD

Nach der Einstellung der erzeugten Parameter kann das virtuelle Gerät gestartet werden. Im Folgenden zeigt der Autor wie Applikationen auf virtuellen Devices installiert werden. Dazu ist vorher ein Überprüfungsschritt notwendig.

Bevor die Applikation auf dem virtuellen Gerät installiert werden kann müssen Einstellungen in der Entwicklungsumgebung Eclipse vorgenommen werden. Dazu muss der Reiter Projekte und dann Einstellungen in Eclipse ausgewählt werden.

In den Einstellungen müssen einige Anpassungen vorgenommen werden. Es ist darauf zu achten, dass die Library „android-support-v7-appcombat“ importiert ist und der Haken bei „Is Library“ nicht gesetzt wurde. Die Library liegt dieser Arbeit bei. Nun kann die Applikation auf dem virtuellen Gerät installiert werden. Dazu muss der Workspace in Eclipse geöffnet

werden. Dafür muss der Reiter Run → Run Configurations geöffnet werden. Folgendes Fenster öffnet sich nun. (siehe Abbildung 19)

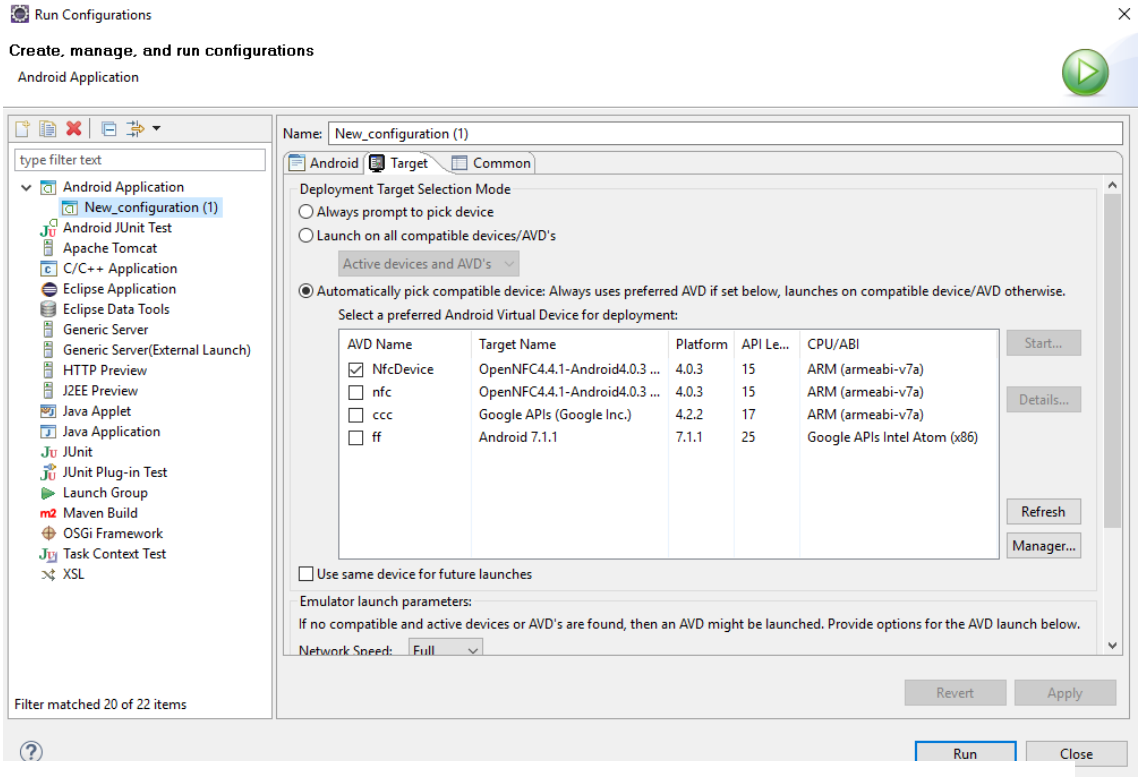

Abbildung 19: Run Configurations Eclipse

Hier muss zunächst das zu installierende Projekt ausgewählt werden. Im nächsten Schritt muss das *Target* also das Zielgerät mit welchem die Applikation ausgeführt werden soll ausgewählt werden. Im beschriebenen Fall also das angelegte virtuelle Gerät. Bei der Bestätigung mit Klick auf den Button *Run* öffnet sich das virtuelle Gerät und installiert die Applikation.

Bevor nun Applikationen getestet werden können muss zunächst die Testumgebung konfiguriert werden. Zuerst muss das AVD korrekt konfiguriert werden. Dazu muss die Software Eclipse im korrekten Workspace geöffnet werden. Um nun das virtuelle Gerät starten zu können muss der Nutzer das Smartphone-Symbol in der oberen Tool-Leiste auswählen. (siehe Abb. 20) Danach öffnet sich ein neues Fenster indem der Nutzer einen Überblick über alle erstellten AVDs erhält. Mit dem Start-Button wird nun das gewünschte Gerät gestartet. Auf dem erstellten AVD ist eine Applikation mit dem Namen „Settings Open NFC“ vorinstalliert. (siehe Abb. 21)

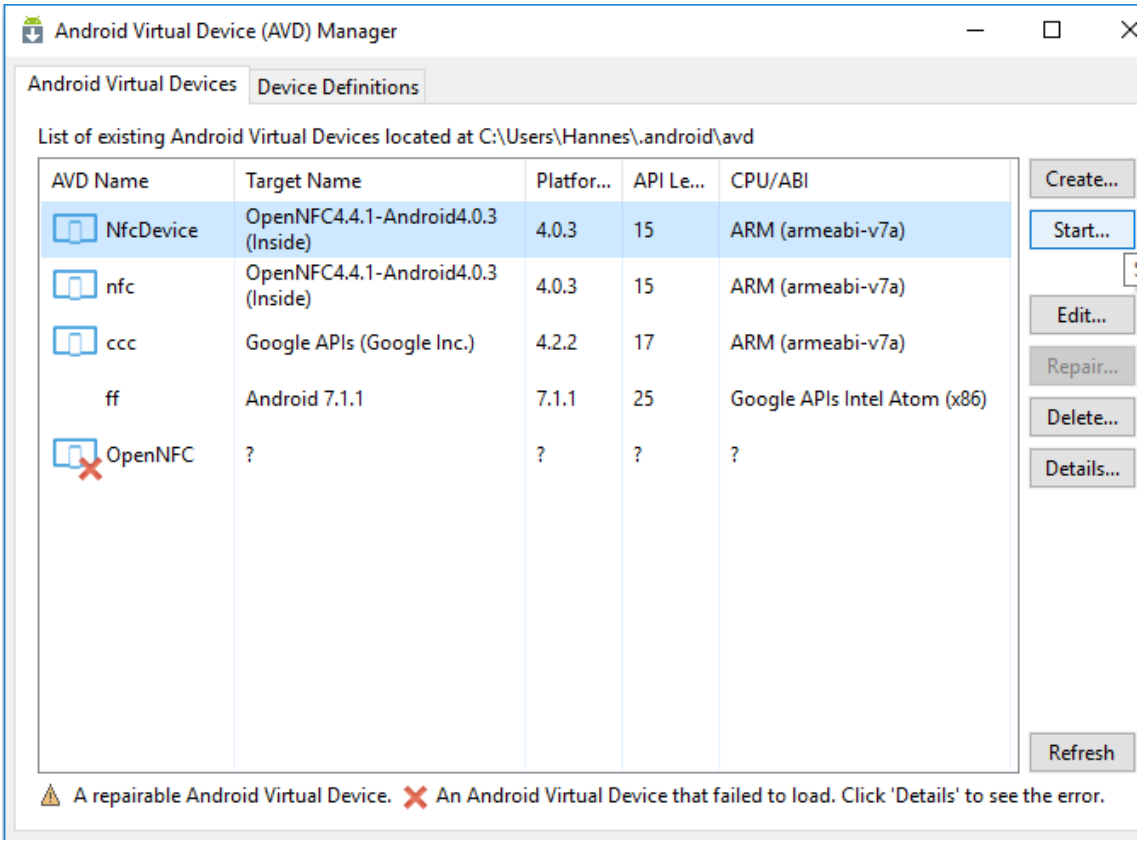

Abbildung 20: Starten des AVDs in Eclipse

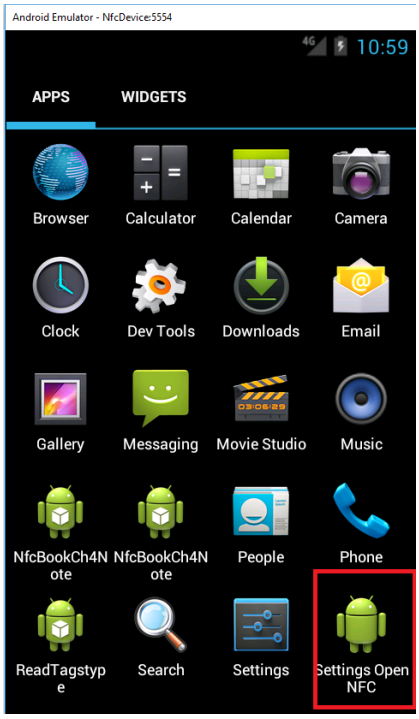

Abbildung 21: AVD NFC-Einstellungen

In dieser App können nun einige Einstellungen vorgenommen werden. (siehe Abb. 22)

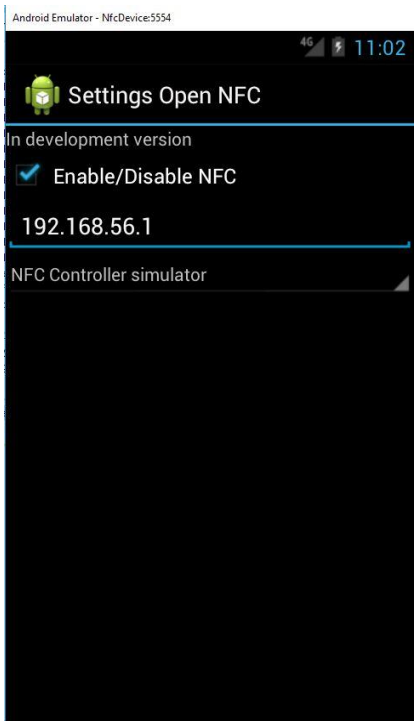

Abbildung 22: AVD NFC-Einstellungen 2

Zuerst muss der Nutzer seine IP-Adresse eintragen. Weiterhin muss im unteren Reiter die Option „NFC Controller Simulator“ ausgewählt und der Haken bei „Enable/Disable NFC gesetzt werden. (siehe Abb. 22) Nach erfolgreichem Abschluss der beschriebenen Schritte ist das Gerät korrekt konfiguriert und verwendbar. Aus Erfahrungen des Autors geht hervor, dass bei manchen Windows-Versionen der Haken erst nach der erfolgreichen Konfiguration aller anderen Tools gesetzt werden kann. Als nächstes muss das Connection Center konfiguriert werden. Dieses ist die Bridge zwischen dem virtuellen Gerät (AVD) und dem NFC-Simulator. Nach der Ausführung der connection_center.exe wird dieses geöffnet. In der rechten Hälfte der unteren Taskleiste kann der Nutzer nun mit der rechten Maustaste auf das grüne Icon klicken und mit der linken Maustaste die Option Show auswählen. (siehe Abb. 23) Danach öffnet sich das Connection Center.

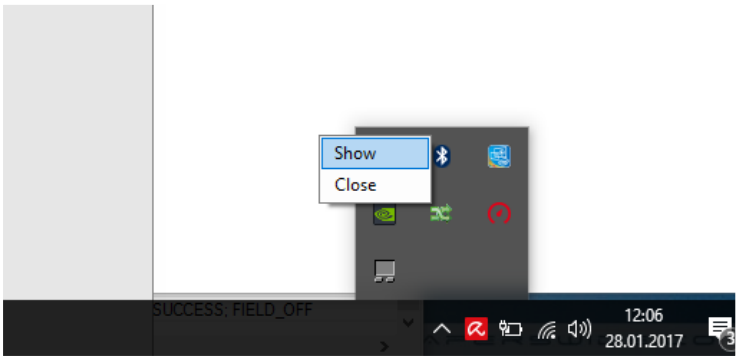

Abbildung 23: Connection-Center-Konfiguration 1

Nun hat der Nutzer die Möglichkeit einige Einstellungen im Connection Center vorzunehmen. Im Reiter Connection sollte darauf geachtet werden, dass folgende Haken gesetzt sind. (siehe Abb.24)

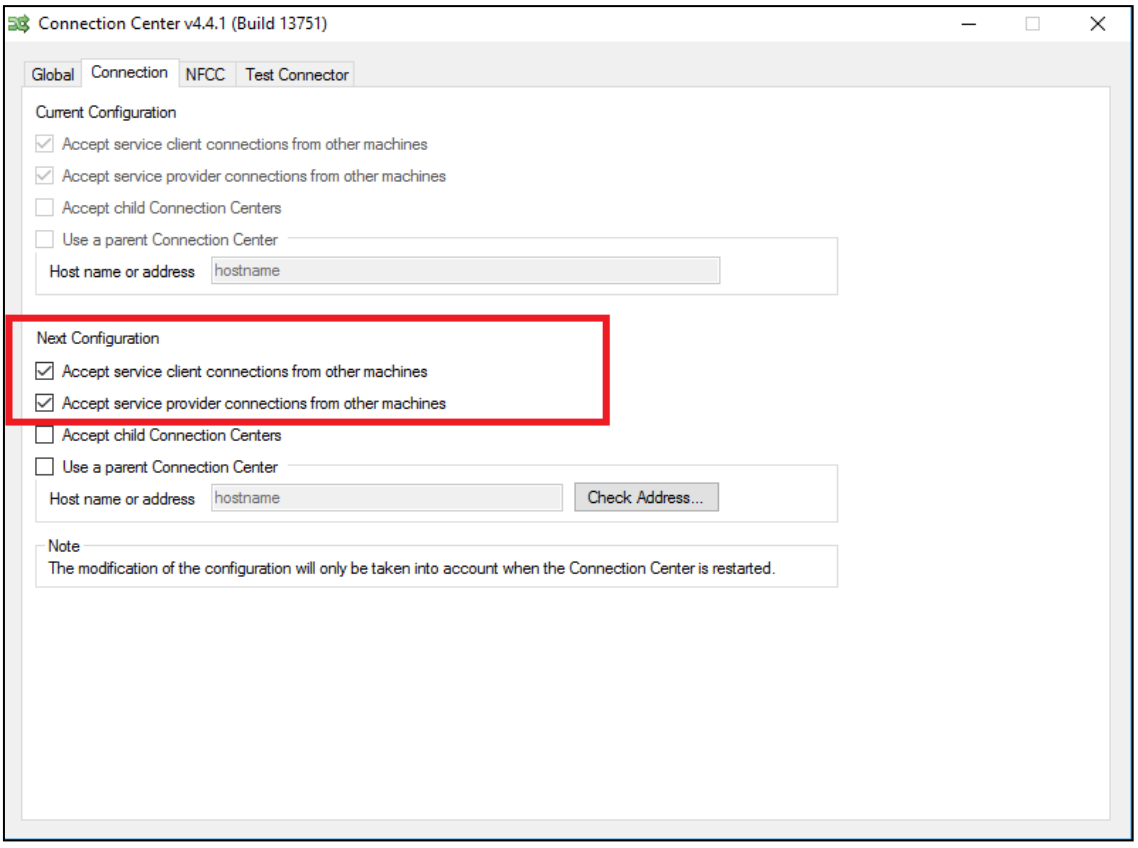

Abbildung 24: Connection-Center-Konfiguration 2

Der verwendete PC und das virtuelle Gerät sind zwei unterschiedliche Maschinen. Durch die Konfiguration des Connection Centers werden eingehende und ausgehende Verbindungen von anderen Maschinen akzeptiert. Im Anschluss ist das virtuelle Gerät in der Lage mit dem Connection Center zu kommunizieren. Falls eine Firewall installiert ist, kann diese Probleme bereiten und die Kommunikation stören. Es sollte grundsätzlich für alle verwendeten Tools eine Ausnahme in der Firewall eingerichtet werden.

4.4.2 Testen einer Applikation in der Testumgebung

Um eine Applikation testen zu können, müssen folgende Schritte exakt befolgt werden. Ansonsten kommt es erfahrungsgemäß zu Kommunikationsproblemen.

1. Starten des Connection Centers, welches sich im Open NFC SDK Edition Ordner befindet. Dort sollte ersichtlich sein das eine eingehende Kommunikation erwartet wird.

2. Starten des NFC Simulators, welcher sich im selben Ordner befindet. Das Connection Center gibt nun die Message „Connected“ aus. (siehe Abbildung 25)

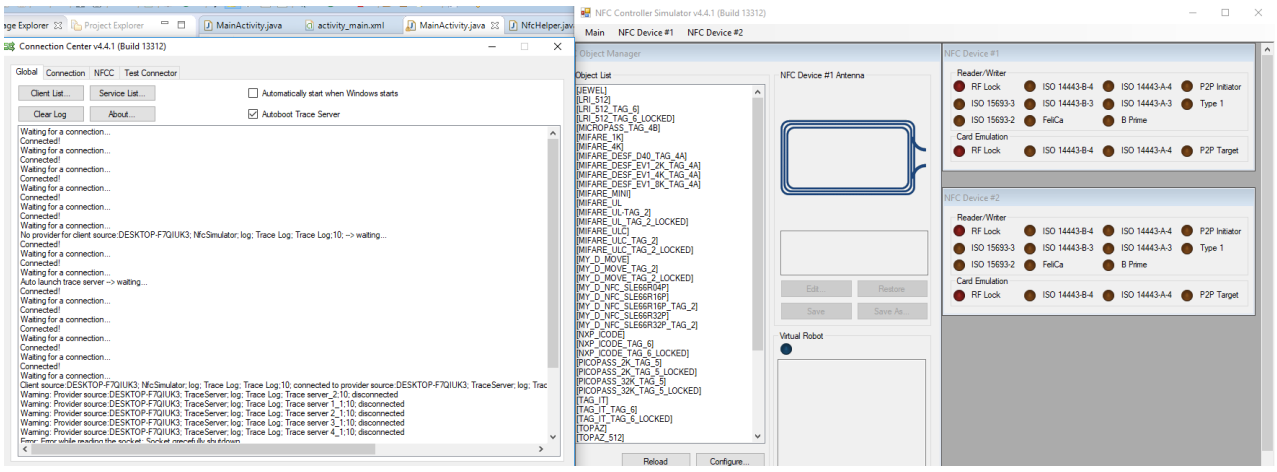

Abbildung 25: Connection Center und NFC-Simulator

3. Der Trace-Server wird nun auch automatisch gestartet.

Es laufen nun das Connection Center, der NFC-Simulator und der Trace Server. Nun muss das zuvor konfigurierte virtuelle Gerät in Eclipse geöffnet werden

Nachdem das virtuelle Gerät geöffnet wurde müssen die Einstellungen in der Applikation „Open NFC Settings“ überprüft werden. (siehe Abbildung 22). Danach zeigt das Connection-Center die bestehende Kommunikation an.

Nun ist die Verbindung hergestellt und die orangenen Lichter im Simulator leuchten. Diese zeigen an, welche Standards das Gerät unterstützt. Nun können wir unsere Applikation ausführen und einen Tag simulieren. (siehe Abbildung 26)

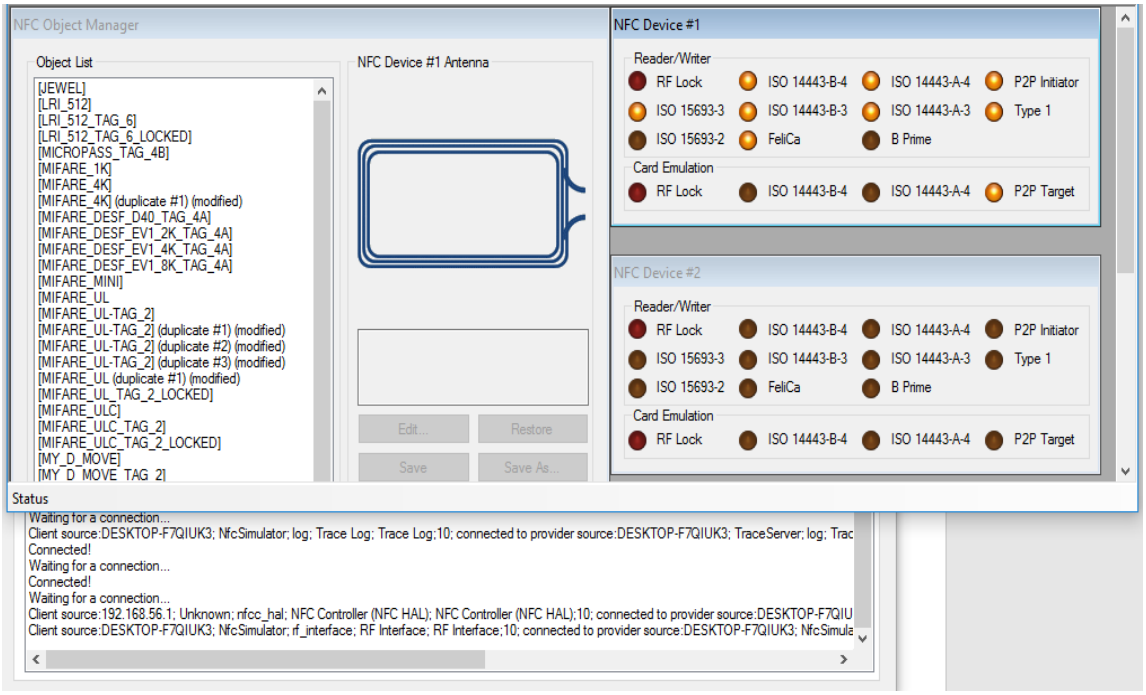

Abbildung 26: *Connection Center und NFC-Simulator 2*

Im folgenden Test startet der Autor die eigene Applikation „*Taginformation*“ und simuliert einen [Mifare-UL-Tag-2] Tag.Nachdem die Applikation ausgeführt wurde, muss der Tag simuliert werden. Nachdem der Tag virtuell „getapped“ wurde liest die Applikation ihn aus. Tappen bezeichnet das heranhalten des Tags an das Lesegerät. Dies wird im Emulator durch einen Doppelklick auf den gewünschten Tag-Typ simuliert. (siehe Abbildung 27)

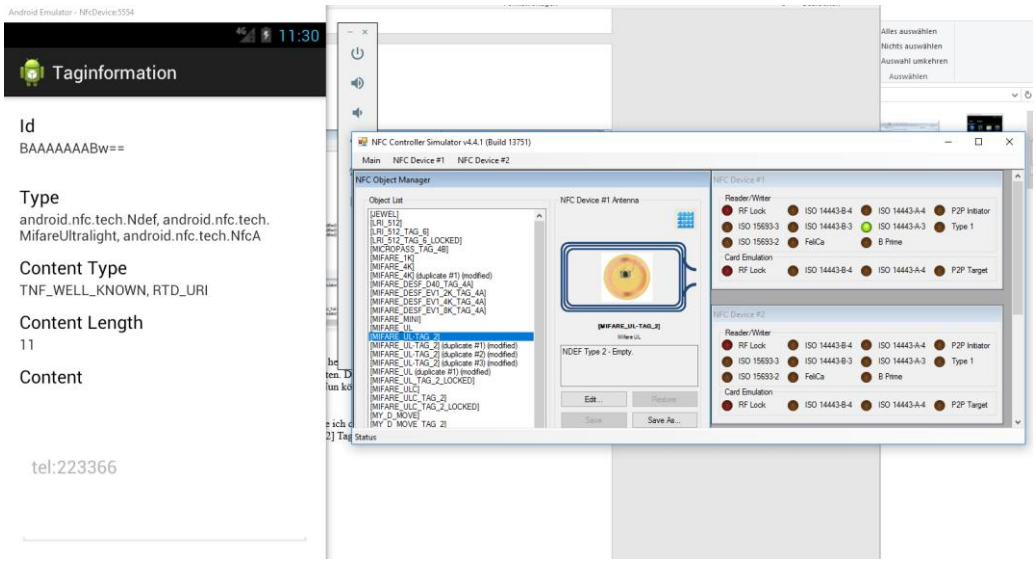

Abbildung 27: Simulation des Tags

4.4.3 Auswahl der zu testenden NFC-Tags

Für die explorative Untersuchung der NFC-Chips empfiehlt sich eine Auswahl von verschiedenen Herstellern, um eine möglichst offene Herangehensweise zu ermöglichen. Alle ausgewählten Chips sollten konform zu den vorgestellten Normen sein und in diversen Infrastrukturen eingesetzt werden. Kapitel 2.1.6 gibt zu den verschiedenen Smartcardtypen- und Herstellern eine Übersicht. Eine Smartcard die in dem zugrundeliegenden Forschungsbereich verwendet werden sollte, ist die MIFARE-Classic-IC. Diese Karte ist in unzähligen Infrastrukturen implementiert und konform zur Norm ISO/IEC 14443. Außerdem eine der ersten Smartcards die von der Firma NXP Semiconductors released wurde. Als zweite Smartcard wurde die nächste Generation von MIFARE-ICs die MIFARE-Ultralight-IC ausgewählt. Diese Smartcard wird weltweit für Einmaltickets in öffentlichen Verkehrsmitteln verwendet und erfüllt ebenfalls den Standard ISO/IEC 14443. Ein weiterer Hersteller ist die Firma Sony. Die sogenannte FeliCa-IC wird in diversen Healthcare-Devices eingesetzt und findet auch in öffentlichen Verkehrsmitteln, vor allem im asiatischen Raum, Anwendung. Weiterhin sollte eine PicoPass-Smartcard nicht fehlen. Diese Karten versprechen höhere Reichweiten und schnellere Kommunikation durch neue Standards. Für dieses Forschungsprojekt wurde die PicoPass-2k-IC ausgewählt. Diese ist konform zur Norm ISO/IEC 14443B und wird unter anderem in diversen Zugangssystemen eingesetzt. Als letzte Smartcard wurde die MIFARE-DESFire-EV1-4k-IC ausgewählt. Diese ist eine der aktuellsten Smartcards und zeigt den derzeitigen Stand der verwendeten Systeme. Außerdem ist auch diese Smartcard konform zur Norm ISO/IEC 14443.

4.4.4 Auswahl der Vergleichsapplikationen

Im Rahmen der Auswahl von vergleichbaren Applikationen müssen zunächst Kriterien festgelegt werden. Aufgrund der Programmierung einer eigenen Android-Applikation hat sich der Autor auch bei den Vergleichsapplikationen auf diese beschränkt. Für einen Vergleich sollten nur Apps aus dem offiziellen Appstore „Google-Playstore“ berücksichtigt werden. Die dort gelisteten Anwendungen müssen signiert sein und werden zumindest einer rudimentären Prüfung unterzogen. Somit ist das Risiko eine schädliche App herunterzuladen wesentlich geringer. [86] Im „Google-Playstore“ existiert mittlerweile ein breites Spektrum an verschiedenen NFC-Apps. Weitere Kriterien sollen die Auswahl einschränken. Um die Applikationen mit der eigenen vergleichen zu können, sollte auch der Basis-Funktionsumfang möglichst identisch sein. Für dieses Forschungsvorhaben wurden vom Autor folgende Funktionen vorausgesetzt. Zum einen soll es möglich sein Metainformationen aus den NFC-Tags auszulesen und diese menschenlesbar darzustellen. Zum anderen sollen die Taginhalte der Tags die auf NDEF basieren ausgelesen und dargestellt werden können. Ein zusätzliches Kriterium für die Vergleichbarkeit ist die Aktualität. In dieser Arbeit wurden nur Applikationen berücksichtigt, die mindestens bis zum 01.01.2016 weiterentwickelt und aktualisiert worden sind. Jeder Entwickler hat grundsätzlich die Möglichkeit den Funktionsrahmen in der Beschreibung selber darzustellen. Um zu überprüfen ob dieser überhaupt erfüllt wird sollten außerdem Nutzerbewertungen berücksichtigt werden. Als letztes Kriterium ist die mindestens benötigte Betriebssystem-Version zu nennen. NFC wurde erst in Android-Version 2.3.3. eingeführt. Für dieses Forschungsprojekt wurden also nur Anwendungen berücksichtigt die als Mindestanforderung diese Android-Version voraussetzen. Der Autor hat nur Applikationen berücksichtigt die mindestens 50 Nutzerbewertungen haben und auf einer Skala von 0-5 Sternen mit mindestens 3,5 Sternen bewertet sind. Die erste Anwendung die ausgewählt wurde ist „DoNFC“ der Firma DOINFOTECH. Diese hat eine Bewertung von 4,2 Sternen und wurde von 56 Benutzern bewertet. Die letzte offizielle Aktualisierung erfolgte am 04. August 2017.

[86] Vgl. Shana, https://support.google.com/googleplay/android-developer/answer/113469?hl=de, zuletzt geprüft am 22.11.2017

Die mindestens benötigte Android-Version ist 3.0. Die aktuelle Version der Applikation Stand 10.10.2017 ist 1.1. Es stehen beide geforderten Basisfunktionen zur Verfügung.[87] Die zweite App die ausgewählt wurde ist „TagInfo" der Firma NXP Semiconductors. Wie in vorherigen Kapiteln ausführlich beschrieben wurde, ist diese Firma eine der leitenden Institutionen in der NFC-Technik. Die Anwendung hat eine Bewertung von 4,0 Sternen erzielt und wurde von 2192 Benutzern bewertet. Die letzte Aktualisierung erfolgte am 24.Mai 2017. Der Benutzer benötigt mindestens Android-Version 2.3.3. Die aktuellste Version der App ist 4.2.3. [88] Die dritte App für dieses Experiment dient als Referenz und ist nicht im „Google-PlayStore" gelistet. Diese wurde vom Autor entwickelt und erfüllt alle geforderten Basisfunktionen. Die mindestens erforderliche Android-Version ist 2.3.3. Die Anwendung wurde bis zum 23.10.2017 weiterentwickelt und aktualisiert. Die aktuelle Version dieser ist 1.3. Tabelle 3 gibt nochmal einen genauen Überblick über alle Applikationen die für diese Arbeit ausgewählt worden sind.

Applikation	Nutzerbewertungen	Mindestanforderung	Letzte Aktualisierung
DoNFC	4,2 Sterne bei 56 Bewertungen	Android-Version 3.0	04.08.2017
TagInfo	4,0 Sterne bei 2192 Bewertungen	Android-Version 2.3.3	24.05.2017
Eigene App	X	Android-Version 2.3.3	23.10.2017

Tabelle 3: Überblick über die Applikationen

[87] Vgl. GooglePlayStore, https://play.google.com/store/apps/details?id=com.doinfotech.donfc&hl=de, zuletzt geprüft am 22.11.2017.

[88] Vgl. Google, https://play.google.com/store/apps/details?id=com.nxp.taginfolite&hl=de, zuletzt geprüft am 22.11.2017

4.4.5 Durchführung des Experiments

Als erstes müssen die virtuellen Smartcards im NFC-Simulator konfiguriert werden. Jeder NFC-Tag wird zweifach abgelegt. Zum einen wird der jeweilige Tag mit einer Telefonnummer beschrieben und zum anderen mit einer URI. So können zwei Formate getestet werden. Die weitere Durchführung des Experiments deckt sich mit der Beschreibung in Kapitel 4.3.2. Zuerst wird die korrekt konfigurierte Testumgebung gestartet. Im Anschluss wird die Applikation ausgewählt die getestet werden soll und auf dem AVD gestartet. Im Anschluss werden die Tags der Reihe nach im Simulator ausgewählt und virtuell an das AVD getapped. Die Logs und Ergebnisse werden übertragen und für die spätere Auswertung gespeichert.

5 Auswertung und Evaluation der Ergebnisse

5.1 Ergebnisse

Zuerst gibt der Autor mit Tabelle 4 einen Überblick über den Funktionsumfang der getesteten Applikationen. Die hier aufgeführten Funktionen sind nur aus den Beschreibungen der Hersteller entnommen und entstammen keinem Praxistest.

Funktion	DoNFC	TagInfo	Eigene App
Auslesen Tag ID	X	X	X
Auslesen Tag Type	X		X
Auslesen Content Type	X	X	X
Auslesen Content Length	X	X	X
Auslesen Content	X	X	X
Beschreiben Tag mit rudimentären Datentypen			X

Tabelle 2: Funktionsumfang der Testapplikationen

Tabelle 5 zeigt die Ergebnisse der Tests mit den Smartcards der MIFARE-Classic-Familie.

Funktion für beschrifteten Tag mit einer URI	DoNFC	TagInfo	Eigene App	Smartcard
Auslesen Tag ID	X	X	X	MIFARE Classic 1k/4k
Auslesen Tag Type	X		X	MIFARE Classic 1k/4k
Auslesen Content Type	X	X	X	MIFARE Classic 1k/4k
Auslesen Content Length	X	X	X	MIFARE Classic 1k/4k
Auslesen Content	X	X	X	MIFARE Classic 1k/4k
Beschreiben des Tags				MIFARE Classic 1k/4k

Funktion für beschrifteten Tag mit einer Tel-Nr:	DoNFC	TagInfo	Eigene App	Smartcard
Auslesen Tag ID	X	X	X	MIFARE Classic 1k/4k
Auslesen Tag Type	X		X	MIFARE Classic 1k/4k
Auslesen Content Type	X	X	X	MIFARE Classic 1k/4k
Auslesen Content Length	X	X	X	MIFARE Classic 1k/4k
Auslesen Content	X	X	X	MIFARE Classic 1k/4k
Beschreiben des Tags			X	MIFARE Classic 1k/4k

Tabelle 5: Ergebnisse MIFARE Classic 1k/4k

Tabelle 6 zeigt die Ergebnisse für MIFARE-Ultralight-Smartcard.

Funktion für beschrifteten Tag mit einer URI	DoNFC	TagInfo	Eigene App	Smartcard
Auslesen Tag ID	X	X	X	MIFARE Ultralight
Auslesen Tag Type	X		X	MIFARE Ultralight
Auslesen Content Type	X	X	X	MIFARE Ultralight
Auslesen Content Length	X	X	X	MIFARE Ultralight
Auslesen Content	X	X	X	MIFARE Ultralight
Beschreiben des Tags				MIFARE Ultralight

Funktion für beschrifteten Tag mit einer Tel-Nr:	DoNFC	TagInfo	Eigene App	Smartcard
Auslesen Tag ID	X	X	X	MIFARE Ultralight
Auslesen Tag Type	X		X	MIFARE Ultralight
Auslesen Content Type	X	X	X	MIFARE Ultralight
Auslesen Content Length	X	X	X	MIFARE Ultralight
Auslesen Content	X	X	X	MIFARE Ultralight
Beschreiben des Tags			X	MIFARE Ultralight

Tabelle 6: Ergebnisse MIFARE Ultralight

Tabelle 7 zeigt die Ergebnisse der Tests mit der FeliCa-IC.

Funktion für beschrifteten Tag mit einer URI	DoNFC	TagInfo	Eigene App	Smartcard
Auslesen Tag ID	X	X	X	FeliCa
Auslesen Tag Type				FeliCa
Auslesen Content Type	X	X	X	FeliCa
Auslesen Content Length	X	X	X	FeliCa
Auslesen Content	X	X	X	FeliCa
Beschreiben des Tags				FeliCa

Funktion für beschrifteten Tag mit einer Tel-Nr:	DoNFC	TagInfo	Eigene App	Smartcard
Auslesen Tag ID	X	X	X	FeliCa
Auslesen Tag Type	X		X	FeliCa
Auslesen Content Type	X	X	X	FeliCa
Auslesen Content Length	X	X	X	FeliCa
Auslesen Content	X	X	X	FeliCa
Beschreiben des Tags			X	FeliCa

Tabelle 7: Ergebnisse FeliCa

Tabelle 8 zeigt die Ergebnisse der Tests mit der PicoPass – IC.

Funktion für beschrifteten Tag mit einer URI	DoNFC	TagInfo	Eigene App	Smartcard
Auslesen Tag ID	X	X	X	PicoPass 2k
Auslesen Tag Type	X		X	PicoPass 2k
Auslesen Content Type	X	X	X	PicoPass 2k
Auslesen Content Length	X	X	X	PicoPass 2k
Auslesen Content	X	X	X	PicoPass 2k
Beschreiben des Tags				PicoPass 2k
Funktion für beschrifteten Tag mit einer Tel-Nr:	DoNFC	TagInfo	Eigene App	Smartcard
Auslesen Tag ID	X	X	X	PicoPass 2k
Auslesen Tag Type	X		X	PicoPass 2k
Auslesen Content Type	X	X	X	PicoPass 2k
Auslesen Content Length	X	X	X	PicoPass 2k
Auslesen Content	X	X	X	PicoPass 2k
Beschreiben des Tags				PicoPass 2k

Tabelle 8: Ergebnisse PicoPass

Tabelle 9 zeigt die Ergebnisse der Tests mit der MIFARE-DESFire-EV1-2k-IC.

Funktion für beschrifteten Tag mit einer URI	DoNFC	TagInfo	Eigene App	Smartcard
Auslesen Tag ID	X	X	X	MIFARE DESFire EV1
Auslesen Tag Type	X		X	MIFARE DESFire EV1
Auslesen Content Type	X	X	X	MIFARE DESFire EV1
Auslesen Content Length	X	X	X	MIFARE DESFire EV1
Auslesen Content	X	X	X	MIFARE DESFire EV1
Beschreiben des Tags				MIFARE DESFire EV1
Funktion für beschrifteten Tag mit einer Tel-Nr:	DoNFC	TagInfo	Eigene App	Smartcard
Auslesen Tag ID	X	X	X	MIFARE DESFire EV1
Auslesen Tag Type	X		X	MIFARE DESFire EV1
Auslesen Content Type	X	X	X	MIFARE DESFire EV1
Auslesen Content Length	X	X	X	MIFARE DESFire EV1
Auslesen Content	X	X	X	MIFARE DESFire EV1
Beschreiben des Tags			X	MIFARE DESFire EV1

Tabelle 9: Ergebnisse MIFARE DESFire EV1 2k

5.2 Beantwortung der Forschungsfragen

Forschungsfrage FF1.1 lautete: Wie viel technisches Knowhow wird benötigt, um Smartcards mit NFC-Chips auszulesen oder zu beschreiben?

Durch das durchgeführte Experiment konnte festgestellt werden, dass jeder mit einem NFC-fähigen Smartphone eine Vielzahl verschiedener Smartcards mit NFC-Chips auslesen kann. Es wird somit nur ein geringes technisches Verständnis für das Auslesen der Daten benötigt. Allerdings können die Ergebnisse oft nicht ohne Fachkenntnisse der Informatik interpretiert werden. Das Beschreiben von Tags gestaltet sich nicht ganz so einfach. Es werden Kenntnisse über NFC, Smartcardtypen und Datentypen benötigt.

Forschungsfrage FF1.2 lautete: Wie viel zeitlicher und finanzieller Aufwand ist nötig, um Smartcards mit NFC-Chips auszulesen oder zu beschreiben?

In diesem Experiment wurden ausschließlich kostenfrei zur Verfügung stehende Applikationen benutzt. Somit ist der finanzielle Aufwand, um Basisinformationen von NFC-Tags auszulesen, gleich Null. Allerdings existieren Apps, die zusätzlich zu den Basisinformationen weitere spezifischere Informationen auslesen können. Ein Beispiel für eine solche App ist „Scheckkartenleser Pro NFC“ von Julien Millau.[89] Diese kostet 5,49€ und weist eine wesentlich höhere Benutzerfreundlichkeit auf. Weiterhin ist es möglich, Anwendungsdaten auszulesen. Dies wird am Beispiel von Kreditkartentransaktionen deutlich gemacht. Es ist also möglich, mit geringem zeitlichen und finanziellen Aufwand Smartcards mit NFC-Chips auszulesen und zu beschreiben. Mehr finanzieller Aufwand bedeutet auch eine Steigerung der Benutzerfreundlichkeit und des Funktionsumfangs.

Forschungsfrage FF1.3 lautet: Existieren Smartcards mit NFC-Chips, die schwieriger auszulesen sind als andere?

Mittlerweile bieten mehrere Hersteller Smartcards mit Verschlüsselungstechnologien an. Damit ist es für Angreifer schwerer, die Kommunikation mitzulesen. Aufgrund der geringen

[89] Vgl. GooglePlaystore, https://play.google.com/store/apps/details?id=com.doinfotech.donfc&hl=de, zuletzt geprüft am 22.11.2017.

Reichweite und des knappen Zeitfensters für das Abgreifen der Daten ist eine verschlüsselte Übertragung der Informationen als sehr sicher einzustufen. Somit existieren Smartcards, die schwieriger auszulesen sind als andere, zum Beispiel die MIFARE-DESFire-EV1-IC und die MIFARE-DESFire-EV2-IC. Eine weitere Schwierigkeit stellt die Kompatibilität dar. Durch NDEF und andere Standards ist es mittlerweile möglich, viele verschiedene Smartcards zu untersuchen, da sie meist auf demselben Prinzip beruhen. Allerdings gibt es noch viele Nischenprodukte, die definitiv schwieriger auszulesen sind, da sie andere Datenformate nutzen. Durch die Beantwortung der untergeordneten Forschungsfragen kann nun auch die übergeordnete beantwortet werden.

Forschungsfrage FF1: Welche Voraussetzungen sind nötig, um Smartcards mit NFC-Chips auszulesen oder zu beschreiben?

Um Smartcards mit NFC-Chips auszulesen oder zu beschreiben wird zunächst ein passendes Lesegerät oder ein Smartphone benötigt. Im Falle des Smartphones gibt es für die am meisten verwendeten Betriebssysteme iOS und Android eine hohe Anzahl an Applikationen zum Auslesen von Smartcards mit NFC-Chips. Je nach Benutzerfreundlichkeit und Funktionsumfang kann auch ein finanzieller Aufwand nötig werden. Für die Interpretation der Ergebnisse der App werden Fachkenntnisse der Informatik benötigt, um die ausgegebenen Daten verstehen zu können. Der Autor hat sich im Rahmen der Forschung dazu entschlossen, eine Applikation selber zu programmieren, um das Verständnis über die Funktionsweise zu erlangen um so die diversen Smartcards mit NFC-Tags zu untersuchen. Für die Programmierung einer solchen App werden Kenntnisse diverser Programmiersprachen und Fachkenntnisse der Informatik vorausgesetzt. Für das Beschreiben der Tags gibt es wesentlich weniger Apps. Dort kommt es auf den gewünschten Datentyp an. Die meisten Anwendungen unterstützen nur wenige. Somit benötigt man Grundkenntnisse der Informatik, um die richtige Applikation auszuwählen und zu nutzen. Bei der Verwendung von speziellen NFC-Lesegeräten wird meist eine individuelle Software benötigt.

5.3 Interpretation der Ergebnisse

Die Ergebnisse zeigen, dass es möglich ist, einen Großteil der Smartcards mit NFC-Chips auszulesen und auch zu beschreiben. Die erste Smartcard, die untersucht wurde, ist die MIFARE-Classic-IC in den Speichergrößen 1024 Byte und 4048 Bytes. (siehe Tabelle 5) Es konnte festgestellt werden, dass die Speichergröße für die Untersuchung unerheblich ist. Die

beiden Karten erzielten dieselben Ergebnisse. Die Tag-ID konnte von allen Testapplikationen ausgelesen werden. Die ID des Tags konnte grundsätzlich von allen getesteten Tags ausgelesen werden, unabhängig vom Hersteller oder des verwendeten Tag-Typs. Bei der MIFARE-Classic-IC konnte außerdem der Content-Type, die Länge des Contents und der Content ausgelesen werden. Dabei spielt es keine Rolle, ob es sich bei den Daten um einfache Datentypen, wie z.B. eine Telefonnummer (String, Double, Long) handelt oder um Tags, die mit dem Datentyp URI beschrieben wurden. Diese Aussagen beziehen sich auf die getesteten Tags und die verwendeten Datentypen. Die Beschreibung der Tags ist nur mit Datentypen wie String, Long oder Double möglich. Dies liegt am Funktionsumfang der programmierten App. Genauere Aussagen werden dazu in der Fehlerbetrachtung getroffen. Exakt dieselben Ergebnisse wie die MIFARE-Classic-IC lieferte die MIFARE-Ultralight-IC (siehe Tabelle 6). Das liegt daran, dass die beiden Smartcards für die Kommunikation dasselbe proprietäre, normkonforme High-Level-Protokoll nutzen. Auch bei dieser Karte war ein Beschreiben des Tags ohne weiteres möglich. Die MIFARE-Ultralight-IC ist dem NFC-Forum-Tag-Typ-2 und die MIFARE-Classic keinem Tagtypen des NFC-Forums zuzuordnen. Damit ist die MIFARE-Classic auch nicht mit vielen Geräten kompatibel. Dennoch konnte diese von allen Applikationen ausgelesen werden. Eine weitere Smartcard, welche getestet wurde, ist die PicoPass-IC mit einer Speichergröße von 2048 Byte (siehe Tabelle 8). Die Ergebnisse für das Auslesen der Tags sind dieselben wie für die MIFARE-Classic-IC und MIFARE-Ultralight-IC. Allerdings konnte die PicoPass-IC nicht beschrieben werden. Es kann nicht abschließend festgestellt werden, warum dies nicht möglich ist. Diese ICs sind konform zum Standard ISO/IEC 14443A und B. Eine Erklärung dafür könnte der eingebaute „Write-once"-Modus sein. Die Hardware kann so vom Hersteller bezogen werden, dass sie nicht wiederbeschreibbar ist.[90] Da keine genaue Dokumentation für die verwendete virtuelle Smartcard vorhanden ist, kann dies aber nicht eindeutig bestimmt werden. Die nächste Smartcard, die getestet wurde, ist die FeliCa von Sony. Es war für keine Applikation möglich, den Tag-Type von der Smartcard auszulesen, die mit dem Datentyp URI beschrieben wurde. Als Erklärung dafür wird nun beschrieben, wie die eigens programmierte App vorgeht. Um sicherzustellen, dass der getappte Tag vom Typ URI ist, testet die App zuerst den TNF und den RDT.

[90] Vgl. PicoPass, http://www.rsmart.com.cn/datasheet/chip_cl/Picopass.pdf, zuletzt geprüft am 04.11.2017

Beide müssten folgende statische Werte enthalten:

- NdefRecord.TNF_WELL_KNOWN
- NdefRecord.RDT_URI

Mit dieser Herangehensweise ist es bei der FeliCa-IC nicht möglich, URIs auszulesen, da schon bei der Erkennung des Tags Fehlermeldungen auftreten. Eine weitere Möglichkeit ist, die Daten auf dieselbe Weise auszulesen, wie dies auch bei Strings der Fall ist. Auf diese Weise könnte man die Daten parsen und den URI erhalten. Eine Umsetzung war im Rahmen dieser Arbeit noch nicht möglich. Die letzte Smartcard, die getestet wurde, ist die MIFARE-DESFire-EV1-IC mit einer Speichergröße von 2048 Byte. Diese IC konnte von allen Applikationen ausgelesen werden. Es handelt sich hierbei um eine IC vom NFC-Forum-Tag-Typ 4. Diese Karte kann mit AES verschlüsselt werden.

5.4 Fazit und Ausblick

Diese Arbeit diente zur Beantwortung der folgenden übergeordneten Forschungsfrage: „Welche Voraussetzungen sind nötig, um Smartcards mit NFC-Chips auszulesen oder zu beschreiben?“. Die Frage resultierte aus einem in der Praxis beobachteten Phänomen, welches wissenschaftlich greifbar gemacht werden sollte. Zu diesem Zweck wurde eine qualitative Untersuchung in Form eines explorativen Experiments durchgeführt. Anhand von verschiedenen Vergleichsapplikationen und einer selbstprogrammierten Referenzanwendung sollten Smartcards mit NFC-Chips auf ihre Auslesbarkeit und Beschreibbarkeit untersucht werden. Dabei hat der Autor eine virtuelle Testumgebung aufgesetzt, um möglichst einheitlich und unter gleichen Bedingungen die Smartcards zu untersuchen. Es hat sich herausgestellt, dass durch die virtuelle Testumgebung einige Störfaktoren nicht berücksichtig werden konnten. Dennoch ist ersichtlich, dass virtuelle Tests für dieses Forschungsgebiet viele Vorteile mit sich bringt. Zum einen konnten durch den NFC-Simulator und die virtuellen Smartcards mehrere verschiedene Karten getestet werden, ohne hohe Kosten zu verursachen. Zum anderen können so weitere Forschungen auf diesen Tests aufbauen und dies mit identischen Bedingungen. Die verwendete Umgebung bietet einen weiteren Vorteil. Durch die Kopplung des Connection-Centers, des virtuellen Gerätes und des NFC-Simulators kann die Kommunikation stets im Log-File des Connection-Centers mitverfolgt und überprüft werden.

Die Ergebnisse zeigen außerdem, dass es mit geringem technischen Knowhow und wenig zeitlichem und finanziellem Aufwand möglich ist, Smartcards mit NFC-Chips auszulesen und zu beschreiben. Die MIFARE-Ultralight-ICs und MIFARE-Classic-ICs werden in vielen kommerziellen Systemen verwendet. Für diese Smartcards war das Auslesen mit allen Testapplikationen erfolgreich. So stellt sich heraus, dass die Daten von den meisten Smartcard-Modellen ohne Vorkenntnisse mitgeschnitten werden können. Der einzige Schutz ist die Entfernung. Diese ist auf maximal 10 cm begrenzt. Innerhalb dieses Radius können die Informationen auf dem Chip im Klartext abgegriffen werden. Diese qualitative Forschung zeigt, dass es keine gute Idee ist, für sensible Anwendungen, wie kontaktlose Bezahlsysteme auf Smartcards, ohne Verschlüsselungstechnik zurückzugreifen. Neuere Modelle, wie die Smartcards der MIFARE-DESFire-Familie, sind mit modernen Verschlüsselungsverfahren ausgestattet, sodass es wesentlich schwerer ist, die Daten abzugreifen. Durch das NFC-Forum und der damit verbundenen Standardisierung von Smartcards sind die Datenformate genormt. Dies ermöglicht Entwicklern und Herstellern eine gemeinsame Weiterentwicklung der Smartcards mit NFC-Chips. Wie die Ergebnisse zeigen, birgt dies aber auch ein enormes Sicherheitsrisiko. Durch die gute Dokumentation ist es für Entwickler auch unkompliziert, Anwendungen zu schaffen, um die Daten auszulesen.

Die entwickelte Android-Applikation zum Auslesen und Beschreiben der Tags erfüllt momentan nur rudimentäre Funktionen und unterstützt nur NDEF-kompatible Tagtypen. Da die meisten Smartcards mit NFC-Chips diesen Standard erfüllen, ist dies für die Zielsetzung dieser Arbeit ausreichend. Bei der Durchführung der Tests wurden einige Verbesserungsmöglichkeiten deutlich. Momentan können Tags mit anderen Datentypen als URI, Long, Double oder String nicht ausgelesen werden. Für die Erkennung eines externen Datentyps könnte eine Prüfung des TNF erfolgen. Dieser gibt folgendermaßen an, ob es sich um einen externen Datentyp handelt:

- NdefRecord.TNF_EXTERNAL_TYPE

Es existieren Datenbanken, die viele externe Datentypen auflisten. Anhand dieser kann geprüft werden, aus welchem Typ der Payload besteht. So kann dieser ausgegeben und analysiert werden. Externe Datentypen sind ähnlich aufgebaut wie URIs. Sie basieren auf dem URN-Schema und sind praktisch URIs mit einem Pointer auf eine Service-Location. Eine weitere Verbesserung ist das Handling von mehreren Tags auf einmal. Befinden sich mehrere Tags in

der Reichweite, wählt die App momentan zufällig einen aus und alle anderen können nicht ausgelesen werden.

Ein zukünftiges Forschungsziel könnte die Übertragung der Android-Applikation auf ein physisches Gerät sein, um so auch Zugriff über andere Hardwarekomponenten zu nutzen. Mittlerweile ist die Akzeptanz für die NFC-Technologie groß. Es ist abzusehen, dass NFC-Anwendungen immer mehr in unseren Alltag integriert werden. Durch die Integration in die meisten Smartphones der neueren Generationen wird der Weg für Massenanwendungen immer mehr geebnet.

Literaturverzeichnis

Literatur:

DELGRANDE, James P.; Faber, Wolfgang (Hg.) (2011): Logic Programming and Nonmonotonic Reasoning: 11th International Conference, LPNMR 2011, Vancouver, Canada, May 16-19, 2011. Proceedings. Berlin, Heidelberg: Springer Berlin Heidelberg.

EGGER, Roman; JOSS, Mario (2010): mTourism. Mobile Dienste im Tourismus. Wiesbaden: Gabler Verlag / Springer Fachmedien Wiesbaden GmbH Wiesbaden. Online verfügbar unter http://www.fritz.eblib.com/patron/FullRecord.aspx?p=748347.

FURHT, Borko (Hg.) (2006): Encyclopedia of Multimedia. Boston, MA: Springer Science+Business Media Inc. Online verfügbar unter http://site.ebrary.com/lib/alltitles/docDetail.action?docID=10134194.

GROß, Jonas (2012): Mifare DESfire - Eine Analyse der Implementierung. Diplomarbeit. Schmalkalden.

LANGER, Josef; ROLAND, Michael (2010): Anwendungen und Technik von Near Field Communication (NFC). Berlin, Heidelberg: Springer-Verlag Berlin Heidelberg. Online verfügbar unter http://site.ebrary.com/lib/alltitles/docDetail.action?docID=10417385.

LINHOFF-POPIEN, Claudia; ZADDACH, Michael; GRAHL, Andreas (Hg.) (2015): Marktplätze im Umbruch. Digitale Strategien für Services im mobilen Internet. Berlin: Springer Vieweg (Xpert.press). Online verfügbar unter http://dx.doi.org/10.1007/978-3-662-43782-7.

MCLLROY, Stuart; ALI, Nasir; HASSAN, Ahmed E. (2016): Fresh apps. An empirical study of frequently-updated mobile apps in the Google play store. In: *Empirical Software Engineering* 21 (3), S. 1346–1370. DOI: 10.1007/s10664-015-9388-2.

PROSSER, Alexander (1993): Standards in Rechnernetzen. Vienna: Springer (Springers Angewandte Informatik). Online verfügbar unter http://dx.doi.org/10.1007/978-3-7091-9273-3.

RANKL, Wolfgang; EFFING, Wolfgang (2008): Handbuch der Chipkarten. Aufbau - Funktionsweise - Einsatz von Smart Cards. 5., überarb. und erw. Aufl. München: Hanser.

SCHIERECK, Dirk; TIELMANN, Artur (2013): Mobile Payment - ein Zahlungsmittel mit Zukunft in Deutschland? In: *WIST* 42 (4), S. 168–174. DOI: 10.15358/0340-1650_2013_4_168.

SUBTIL, Vitor (2014): Near field communication with Android cookbook. Discover the endless possibilities of using Android NFC capabilities to enhance your apps over 50 practical recipes. Birmingham, UK: Packt Pub (Quick answers to common problems). Online verfügbar unter http://proquest.tech.safaribooksonline.de/9781783289653.

VAN TILBORG, Henk C. A.; JAJODIA, Sushil (Hg.) (2011a): Encyclopedia of Cryptography and Security. 2. ed. Boston, MA: Springer Science+Business Media LLC (Springer reference). Online verfügbar unter http://dx.doi.org/10.1007/978-1-4419-5906-5.

VAN TILBORG, Henk C. A.; JAJODIA, Sushil (Hg.) (2011b): Encyclopedia of Cryptography and Security. 2. ed. Boston, MA: Springer Science+Business Media LLC (Springer reference). Online verfügbar unter http://dx.doi.org/10.1007/978-1-4419-5906-5.

ZEFFERER, Thomas (2012): Konzepte und Umsetzungen NFC - Basierter Zahlungssysteme. In: Zentrum für sichere Informationstechnologie - *Austria*, 10.07.2012 (1.0)

SEMICONDUCTORS, NXP (2009): MIFARE Ultralight C. Product Sheet. Online verfügbar unter https://www.nxp.com/docs/en/data-sheet/MF0ICU2_SDS.pdf

SEMICONDUCTORS, NXP (2016): MIFARE DESFire EV2. Fact Sheet. Online verfügbar unter https://www.nxp.com/docs/en/fact-sheet/MIFARE-DESFIRE-EV2-FS.pdf

MARTIN H., WEIK D. (2001): Computer Science and Communication Dictionary. Springer-Verlag Berlin Heidelberg. Online verfügbar unter https://link.springer.com/reference-work/10.1007%2F1-4020-0613-6

FREED N., BORENSTEIN N. (1996): Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types

Online-Quellen:

SCHIRRMACHER, Dennis (2015): Smartphone - Trojaner zapft NFC-Kreditkarte an. Unter Mitarbeit von Dennis Schirrmacher. Hg. v. Heise Medien. Online verfügbar unter https://www.heise.de/security/meldung/Smartphone-Trojaner-zapft-NFC-Kreditkarte-an-2674454.html, zuletzt geprüft am 12.11.2017.

Credit Card Glossary: Terms and Definitions. Dual Interface Cards. Online verfügbar unter https://www.creditcards.com/glossary//term-dual-interface-chip-card.php, zuletzt geprüft am 08.10.2017

Transport for London, Every Journey Matters (Hg.): Media. Online verfügbar unter https://tfl.gov.uk/info-for/media/?cid=pp017, zuletzt geprüft am 06.10.2017.

Transport for London, Every Journey Matters: What is Oyster? Online verfügbar unter https://tfl.gov.uk/fares-and-payments/oyster/what-is-oyster?intcmp=1685, zuletzt geprüft am 17.10.2017.

Frauenhofer SIT: RFID - Sicherheit und Angriffsmethoden. Online verfügbar unter http://www.rfid-basis.de/rfid-sicherheit.html, zuletzt geprüft am 17.10.2017.

Google TagInfo. Hg. v. Google. Online verfügbar unter https://play.google.com/store/apps/details?id=com.nxp.taginfolite&hl=de, zuletzt geprüft am 22.11.2017.

GooglePlayStore. Hg. v. Google. Online verfügbar unter https://play.google.com/store/apps/details?id=com.doinfotech.donfc&hl=de, zuletzt geprüft am 22.11.2017.

Java, About Hg. V. Java. https://java.com/about, zuletzt geprüft am 27.09.2017

MILINKOVIC, M. Nov: About the Eclipse Foundation. Online verfügbar unter http://www.eclipse.org/org/#about, zuletzt geprüft am 27.09.2017.

ALMEIDA, Marcio: Hacking Mifare Classic Cards. Online verfügbar unter https://www.blackhat.com/docs/sp-14/materials/arsenal/sp-14-Almeida-Hacking-MIFARE-Classic-Cards-Slides.pdf, zuletzt geprüft am 17.10.2017.

Android Developers, Ndef. Online verfügbar unter https://developer.android.com/reference/android/nfc/tech/Ndef.html, zuletzt geprüft am 16.10.2017.

NFC-Forum, What it Does. Hg. v. NFC-Forum. Online verfügbar unter https://nfc-forum.org/what-is-nfc/what-it-does/, zuletzt geprüft am 12.11.2017.

REWE, Pressemitteilung kontaktloses zahlen. Hg. v. Presseabteilung REWE. Online verfügbar unter https://presse.rewe.de/artikel/rewe-bietet-kontaktloses-zahlen-an/, zuletzt geprüft am 10.11.2017.

Shana, App hochladen. Hg. v. Google. Online verfügbar unter https://support.google.com/googleplay/android-developer/answer/113469?hl=de, zuletzt geprüft am 22.11.2017.

Sony Global, FeliCa Contactless IC Card System. Hg. V. Sony Global Online verfügbar unter https://www.sony.net/Products/felica/about/scheme.html, zuletzt geprüft am 10.10.2017

KURI,Jürgen: Der Internetausweis. Online verfügbar unter https://www.heise.de/ct/artikel/Der-Internet-Ausweis-1111003.html, zuletzt geprüft am 05.09.2017

HOLZINGER, Michael: Near Fiel Communication Forum gegründet. Online verfügbar unter http://www.wcm.at/contentteller.php/news_story/near_field_communication_forum_gegruendet.html, zuletzt geprüft am 07.09.2017

NearFieldCommunication.org, NFC Signaling Technologies. Online verfügbar unter http://nearfieldcommunication.org/nfc-signaling.html, zuletzt geprüft am 20.09.2017

Iso.org, ISO/IEC 14443-3:2011-Identification cards-Contactless integrated circuit cards - Proximity cards-Part 3: Initialization and anticollision. Online verfügbar unter https://www.iso.org/standard/50942.html, zuletzt geprüft am 20.09.2017

Eigenständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Literatur und Hilfsmittel angefertigt habe. Stellen, die wörtlich oder sinngemäß aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit wurde in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegt.

Ort, Datum Vorname Nachname